

(19)世界知的所有権機関
国際事務局(43)国際公開日
2005年10月20日 (20.10.2005)

PCT

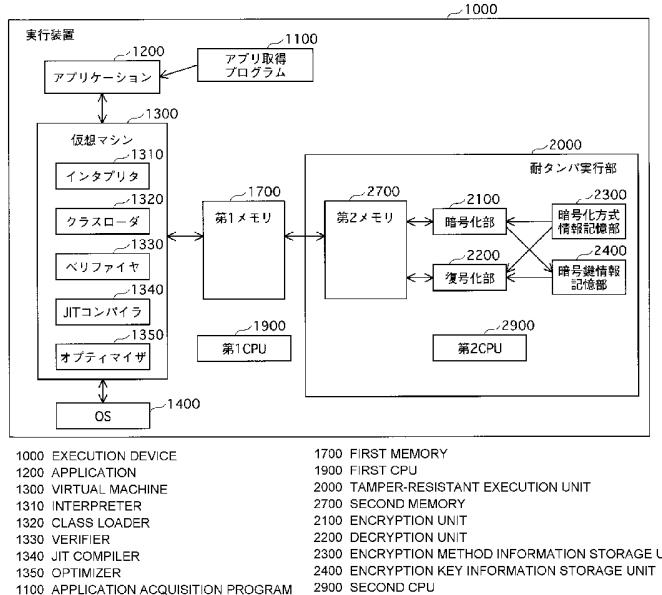
(10)国際公開番号
WO 2005/098570 A1

- (51) 国際特許分類⁷: G06F 1/00, 9/44, 12/14
 (21) 国際出願番号: PCT/JP2005/006307
 (22) 国際出願日: 2005年3月31日 (31.03.2005)
 (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2004-110781 2004年4月5日 (05.04.2004) JP
 (71) 出願人(米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO.,LTD.) [JP/JP]; 〒5718501 大阪府門真市大字門真1006番地 Osaka (JP).
- (72) 発明者; および
 (75) 発明者/出願人(米国についてのみ): 黒瀧満 (KURO-TAKI, Mitsuru). 中村智典 (NAKAMURA, Tomonori).
 (74) 代理人: 中島司朗, 外 (NAKAJIMA, Shiro et al.); 〒5310072 大阪府大阪市北区豊崎三丁目2番1号淀川5番館6F Osaka (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, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE,

[続葉有]

(54) Title: EXECUTION DEVICE

(54) 発明の名称: 実行装置



1000 EXECUTION DEVICE	1700 FIRST MEMORY
1200 APPLICATION	1900 FIRST CPU
1300 VIRTUAL MACHINE	2000 TAMPER-RESISTANT EXECUTION UNIT
1310 INTERPRETER	2100 ENCRYPTION UNIT
1320 CLASS LOADER	2200 DECRYPTION UNIT
1330 VERIFIER	2300 ENCRYPTION METHOD INFORMATION STORAGE UNIT
1340 JIT COMPILER	2400 ENCRYPTION KEY INFORMATION STORAGE UNIT
1350 OPTIMIZER	2900 SECOND CPU
1100 APPLICATION ACQUISITION PROGRAM	

WO 2005/098570 A1

(57) Abstract: An execution device executes an application program having a plurality of parts constituted by an instruction, generates a native code from the part in a particular memory area, and encrypts the native code generated so as to create an encrypted code. After this, the execution device deletes the native code generated and stores the created encrypted code in encrypted code storage means. When executing each part, the execution device decrypts the encrypted code corresponding to the part stored in the encrypted code storage means, executes the native code, and deletes the native code which has been executed.

(57) 要約: 実行装置は、命令で構成される部分を複数個有するアプリケーションプログラムを実行し、部分からネイティブコードを特定のメモリ領域に作成し、作成したネイティブコードを暗号化して暗号コードを作成した後、前記作成したネイティブコードを削除し、作成した暗号コードを暗号コード記憶手段に記憶し、各部分を実行する際に、暗号コード記憶手段に記憶されている当該部

[続葉有]



SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

- (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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR),

添付公開書類:
— 国際調査報告書

2文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明細書

実行装置

技術分野

[0001] 本発明は、プログラムの盗聴および改竄を防止する技術に関し、特に、プログラム実行時に、実行中のプログラムが盗聴および改竄されることを防止する技術に関する。

背景技術

[0002] 近年、様々なアプリケーション・プログラム(以下、「アプリケーション」という。)が、パソコンをはじめ、デジタルテレビや携帯電話機などの情報処理機能を有した装置で実行されている。

これらのアプリケーションは、元々その装置に組み込まれていたものや、ユーザが購入し乗せたもの、また、プログラム配信サービスによって提供されているものなど様々である。プログラム配信サービスによるものとは、例えば、インターネットを介してダウンロードしたり、デジタル放送の放送波に多重化されて送信されてくるものである。

[0003] その一方で、プログラムが改竄されたり、データが盗まれたりという不正行為が起こっている。このような不正行為は、プログラム配信サービス等の円滑な実施の障害となることから、プログラムの改竄等を防止する技術が開発されている。

例えば、アプリケーションを仮想マシンで実行するような実行装置においては、アプリケーションを暗号化などにより保護したとしても、仮想マシン内では、復号化された平文で処理を行う必要があるため、プログラムを盗聴、改竄される可能性がある。そこで、Java(登録商標)言語におけるバイトコードに相当するような中間コードである命令セットを独自に作成し、かつ公開も行わないこととして、仮想マシン内でのアプリケーションを保護しようとする技術が開発されている(特許文献1参照)。この技術によれば、仮想マシン内の中間コードは、盗聴、改竄から守られる。

特許文献1:特開2002-132364号公報

発明の開示

発明が解決しようとする課題

- [0004] しかし、アプリケーションの実行速度を考慮すると、中間コードを順番に解釈し、実行していくよりも、CPU(Central Processing Unit)が直接実行できるネイティブコードを実行するほうが速いため、実行速度が要求されるアプリケーションを実行する仮想マシンでは、中間コードで作成されたアプリケーションをロード等するときにネイティブコードに変換し、次からはネイティブコードを実行する方式が取られる場合が多い。
- [0005] この場合、ネイティブコードは、通常、必要がなくなるまでアクセス可能なメモリ上に置かれる場合が多いにも関わらず、何ら盗聴、改竄から守る方策は取られていないことになる。

そこで、本発明は、中間コードのみならず、ネイティブコードの盗聴および改竄を防止するような実行装置の提供を目的とする。

課題を解決するための手段

- [0006] 上記課題を解決する為に、本発明の実行装置は、命令で構成される部分を複数個有するアプリケーションプログラムを実行する実行装置であって、部分からネイティブコードを特定のメモリ領域に作成し、作成したネイティブコードを暗号化して暗号コードを作成した後、前記作成したネイティブコードを削除し、作成した暗号コードを暗号コード記憶手段に記憶するネイティブコード作成手段と、暗号コード記憶手段に記憶されている暗号コードを復号し、ネイティブコードを特定のメモリ領域に作成する復号手段と、各部分を実行する際に、当該部分に該当する暗号コードを前記復号手段により復号し、ネイティブコードを実行した後、実行したネイティブコードを削除する実行制御手段とを備えることを特徴とする。

発明の効果

- [0007] 本発明に係る実行装置は、上述の構成を備えることにより、仮想マシン内でのネイティブコードは暗号化されており、実行する時に、実行する範囲のみが復号された状態となるので、第三者からの盗聴、改竄される機会を大幅に減らすことができるようになる。

従って、仮想マシンで実行されるアプリケーションの盗聴および改竄を、大幅に防
止できることとなる。

- [0008] また、前記アプリケーションプログラムは、更に、各部分ごとに暗号化するか否かを

示す秘匿性情報を含み、前記ネイティブコード作成手段および前記実行制御手段は、前記秘匿性情報が暗号化することを示している部分に限り、当該部分を対象として処理することとしてもよい。

これにより実行装置は、アプリケーションの秘匿性が高い場合のみ、暗号化することができる、実行速度を落とさずに、アプリケーションを盗聴、改竄から守ることが出来るようになる。

[0009] また、前記秘匿性情報は、更に、秘匿の度合いを示す情報を含み、前記ネイティブコード作成手段は、前記秘匿性情報が暗号化することを示している場合に限り、前記秘匿性情報に基づいて決定された暗号化方式で暗号化した暗号コードを作成することとしてもよい。

これにより実行装置は、アプリケーションの秘匿性に応じて、暗号化方式を選ぶことができ、秘匿性の高い部分は、より暗号強度の高い暗号化方式で暗号化することができる、秘匿性の高い部分は、より強固に改竄等から守る事が出来るようになる。また、暗号強度の高い暗号化方式で暗号化する部分を選択することにより、全体としての実行速度の低下を抑えることができる。

[0010] また、前記部分は、分岐命令を含まない一連の命令群であることとしてもよい。

また、前記アプリケーションプログラムは、オブジェクト指向言語で作成され、前記部分は、メソッドであることとしてもよい。

また、前記部分は、1つの命令であることとしてもよい。

これにより実行装置は、保護したい部分を、より細かな単位で暗号化することができ、仮想マシン内での復号化されるネイティブコードの量を、必要最小限度にし、部分の実行時間も必要最小限度にすることができるので、よりきめ細かくアプリケーションを盗聴、改竄から守ることが出来るようになる。

[0011] また、本発明の実行装置は、命令で構成される部分を複数個有するアプリケーションプログラムを実行する実行装置であって、所定のメモリ領域への読み出し及び書き込みのアクセスを、許可又は禁止するよう制御するアクセス制御手段と、前記アクセス手段により、前記所定のメモリ領域への書き込みのアクセスを許可し、部分からネイティブコードを前記所定のメモリ領域に作成し、前記アクセス手段により、前記所定のメモリ領

域への書き込みのアクセスを禁止するネイティブコード作成手段と、部分を実行する際に、前記アクセス手段により、前記所定のメモリ領域への読み込みのアクセスを許可し、当該部分に該当するネイティブコードを読み出した後、前記アクセス手段により、前記所定のメモリ領域への読み込みのアクセスを禁止する実行制御手段とを備えることを特徴とする。

- [0012] 本発明に係る実行装置は、上述の構成を備えることにより、ネイティブコードはアクセス不可能の領域に記憶しておき、実行するときにのみ読み出すことができるので、第三者からの盗聴、改竄される機会を大幅に減らすことができるようになる。

図面の簡単な説明

[0013] [図1]本発明にかかる実行装置の構成を表す図である。

[図2]クラスファイルの構成例を示す図である。

[図3]クラスブロックの構成例を示す図である。

[図4]図4(a)は、クラスヘッダの構成例を示す図であり、図4(b)は、メソッドヘッダの構成例を示す図である。

[図5]図5(a)は、暗号強度情報の内容例を表す図であり、図5(b)は、分割情報の構成例および内容例を表す図である。

[図6]暗号化方式情報の構成例および内容例を表す図である。

[図7]暗号鍵情報の構成例および内容例を表す図である。

[図8]実行装置1000の処理を表すフローチャートである。

[図9]クラスロード処理を表すフローチャート、および第1メモリの内容を表す図である。

[図10]ベリファイ処理を表すフローチャート、および第1メモリの内容を表す図である。

[図11]メソッド実行処理を表すフローチャート、および第1メモリの内容を表す図である。

[図12]JITコンパイル処理を表すフローチャート、および第1メモリの内容を表す図である。

[図13]最適化処理を表すフローチャート、および第1メモリの内容を表す図である。

[図14]暗号化処理を表すフローチャート、および第1メモリ、第2メモリの内容を表す

図である。

[図15]復号化処理を表すフローチャート、および第1メモリ、第2メモリの内容を表す図である。

[図16]実施形態2のメモリの構成を表す図である。図16(a)は、メモリ上の読み込み可能属性を設定した状態を表す図であり、図16(b)は、メモリ上の読み込み不可能属性を設定した状態を表す図である。

[図17]従来技術の構成を表す図である。

符号の説明

[0014]	201	プログラム
	201	暗号化プログラム
	202	復号化手段
	203	仮想機械
	204	CPU
	1000	実行装置
	1100	アプリ取得プログラム
	1200	アプリケーション
	1210	クラスファイル
	1300	仮想マシン
	1301	暗号化方式情報
	1310	インタプリタ
	1320	クラスローダ
	1330	ベリファイヤ
	1340	JITコンパイラ
	1350	オプティマイザ
	1400	OS
	1701	クラスブロック
	1700	第1メモリ
	1710	クラスヘッダ

1720 暗号強度情報
1730 分割情報
1800 メソッド
1810 メソッドヘッダ
1811 暗号強度情報
1850 コード
1900 第1CPU
2000 耐タンパ実行部
2100 暗号化部
2200 復号化部
2300 暗号化方式情報記憶部
2301 暗号化方式情報
2310 鍵ID
2320 暗号鍵
2330 復号鍵
2340 暗号アルゴリズム
2400 暗号鍵情報記憶部
2401 暗号鍵情報
2700 第2メモリ
2900 第2CPU
3700 メモリ

発明を実施するための最良の形態

[0015] <実施形態1>

<概要>

本発明に係る実行装置は、アプリケーションを実行する際には、プログラムの分岐命令などの存在により、すべての命令が順次実行されていくものではないこと、また、仮想マシンでアプリケーションを実行する場合には、ベリファイの処理や、最適化の処理など、複数の処理を経た後に、初めて実行されるものであることに着目して、ア

プリケーションを盗聴、改竄から守ろうとするものである。

- [0016] すなわち、アプリケーションが暗号化されていたとしても、各処理を行うには、いわゆる平文の状態でメモリ上に置かざるを得ない場合があり、コスト面などを考慮すると、このメモリは耐タンパ性を有してはいないのが通常である。

従って、このメモリ上に置かれる平文の量を出来るだけ少なくし、かつ、平文である時間を出来るだけ短くすることで、悪意ある第三者による盗聴、改竄の機会を減らそうというのが、本発明の狙いである。

- [0017] 具体的には、例えば、Java(登録商標)言語で作成されたアプリケーションであれば、クラスのロード時に、クラスを複数に分割し、それぞれの分割部分を暗号化して、メモリ上に配置する。

その後、ベリファイ処理やJITコンパイル処理などを行う際に、原則として、1つの分割部分のみを復号し処理を行い、処理を施して得られた新しい部分は暗号化し、処理の基となつた部分は削除する。この手順を繰り返すことにより、1つのクラスに対する処理を行う。

- [0018] このことにより、常に、平文であるバイトコードまたはネイティブコードは、処理に必要な最小の量が、最短時間の間だけメモリ上に存在することになり、平文のプログラムが盗聴、改竄されることを可能な限り防止することが可能となる。

すなわち、分割・暗号化されたデータを必要なときに必要な分だけ復号化することで、重要なデータが復号化された状態でアクセス可能なメモリに格納される時間を削減することが可能になるのである。

- [0019] また、本発明は、アプリケーションの重要度に応じて、自由に暗号化の度合いを変えることが可能であることも特徴とする。

アプリケーションの秘匿性の度合いは、そのアプリケーションを作成するときに、作成者またはそのユーザのみが知りうるものである。

従って、本発明に係る実行装置は、クラス内の特定の分割部分には、暗号の強さを指定できるものであり、秘匿性が低く暗号化を行わなくてもいい部分と、比較的秘匿性が高い部分などで、暗号化の方法をかえることができる。また、それぞれの分割部分に、異なる暗号鍵を用いて暗号化を施すことで、全体の暗号強度を高めることができる。

きる。

- [0020] 結果として、暗号化、復号化にかかる時間を最小限度にすることができるので、全体としての実行時間の低下を最小限に抑えつつ、実行中のプログラムの改竄等を防ぐ機能を備えているものであるといえる。

以下、本発明の実施形態に係る実行装置について説明する。

本実施形態では、Java(登録商標)バーチャルマシン上で動くJava(登録商標)アプリケーションについて説明する。

- [0021] <構成>

図1は、本発明にかかる実行装置の構成を表す図である。

実行装置1000は、アプリ取得プログラム1100、アプリケーション1200、仮想マシン1300、OS(Operating System)1400、第1メモリ1700、第1CPU(Central Processing Unit)1900および耐タンパ実行部2000で構成される。

- [0022] 実行装置1000は、アプリケーションを実行する機能のほか、それぞれの装置特有の機能を有する(図示していない)。本実行装置1000は、具体的には、デジタルテレビ、セットトップボックス、DVDレコーダー、BD(Blu-ray Disc)レコーダー、カーナビ端末、携帯電話機、PDA(Personal Digital Assistance)などの、Java(登録商標)仮想マシンを搭載する電子機器全般が該当する。

- [0023] ここで、本実行装置1000のアプリケーションを実行する機能とは、通常のパーソナルコンピュータやデジタル家電機器等に搭載されているソフトウェア実行手段と同様のものである。例えば、実行装置1000がデジタルテレビであれば、受信したデジタルデータを画像に変換し表示するアプリケーションを実行することになる。

まず、アプリケーション1200は、本実行装置1000で実行するアプリケーションであって、本装置外のアプリケーションファイルからダウンロードされたものであるとする。

- [0024] アプリ取得プログラム1100は、Java(登録商標)言語で記述されており、アプリケーション1200を外部のファイルから読み込み、アプリケーションの実行に必要な処理を実行、制御する機能を有する。

次に、仮想マシン1300は、Java(登録商標)言語で記述されたプログラムを逐次解析し実行するJava(登録商標)仮想マシンである。言い換えれば、ソフトウェアプログ

ラムである仮想マシン1300が、仮想のCPUをシミュレートして、Java(登録商標)の命令コードを解析実行する。

- [0025] この仮想マシン1300は、インタプリタ1310、クラスローダ1320、ベリファイヤ1330、JITコンパイラ1340およびオプティマイザ1350で構成される。

インタプリタ1310は、アプリケーション1200のバイトコードを解釈、実行する機能を有し、仮想マシンにおいて中核な処理を行う。具体的には、プログラムで呼び出されたメソッドが、まだローディングされていなければ、そのメソッドが属するクラスをロードするよう指示を出したり、メソッドの実行を指示したりする。ここで、バイトコードとは、Java(登録商標)言語で書かれたソースをコンパイルすることにより得られる、ハードウェアに依存しない中間コードである。

- [0026] クラスローダ1320は、アプリケーション1200を構成するクラスファイルを、外部のファイルから読み込みロードする機能を有する。また、クラスローダ1320は、クラスをアンロードする機能も有する。実行が終了し不要になったクラスを仮想マシン1300から取り除く機能である。

また、ベリファイヤ1330は、クラスのデータ形式の不備や、クラスに含まれるバイトコードの安全性を判定する機能を有する。クラスローダ1320は、ベリファイヤ1330において妥当ではないと判定されたクラスのロードは行わない。

- [0027] JITコンパイラ1340は、バイトコードを第1CPU1900が理解可能な実行形式に翻訳する機能を有する。

オプティマイザ1350は、プログラムの仕様を変えずに、より高速に実行でき、より小さいメモリ使用量で動作するように、プログラムの細かな見直しを行う機能を有する。

次に、OS1400は、他のサブプログラムを平行して実行するカーネル及び、ライブラリで構成される技術の総称であり、仮想マシン1300をサブプログラムとして実行する。例えば、Linux等がある。

- [0028] 第1メモリ1700は、いわゆる作業メモリであり、RAM(Random Access Memory)で構成される。具体的には、SRAM(Static Random Access Memory)、DRAM(Dynamic Random Access Memory)等の一次記憶メモリである。

この第1メモリは、第1CPU1900が使用し、OSや仮想マシンなども、この第1メモリ

にロードされて実行されていることになる。

- [0029] 第1CPU1900は、仮想マシン1300、OS1400、アプリケーション1200等を実行する機能を有する。

実行装置1000は、他に、仮想マシン1300のプログラムなどを記憶しておく手段(図示していない)を有している。例えば、ROM(Read Only Memory)、具体的にはフラッシュメモリやハードディスク等の不揮発性メモリである。また、BD-ROM等の記録媒体であってもよい。

- [0030] 次に、耐タンパ実行部2000は、暗号化部2100、復号化部2200、暗号化方式情報記憶部2300、暗号鍵情報記憶部2400、第2メモリ2700および第2CPU2900で構成される。

耐タンパ実行部2000は、悪意のある第三者からの攻撃を防御しつつ安全にプログラムを実行させることができるプログラム実行手段である。

- [0031] 暗号化部2100は、仮想マシン1300から、第1メモリ1700上のアドレスを受け取り、そのデータを暗号化する機能を有する。また、どのような暗号方式で暗号化するかを決定する機能も有する。

また、復号化部2200は、仮想マシン1300から、第1メモリ1700上のアドレスを受け取り、そのデータを復号化する機能を有する。

- [0032] 暗号化方式情報記憶部2300は、本実行装置で使用する暗号化方式が記憶されている。具体的には、複数の暗号アルゴリズムと鍵の組合せであり、本実施形態では、6種類の組合せが記憶されている。図6を用いて、後で説明する。

次に、暗号鍵情報記憶部2400は、第1メモリ上のデータのアドレスと、そのデータをどの暗号化方式で暗号化したかを記憶している。ここでの暗号化方式は、暗号化方式情報記憶部2300に記憶されている暗号化方式のうちのいずれかである。図7を用いて、後で説明する。

- [0033] 第2メモリ2700は、第1メモリ1700と同様の、いわゆる作業メモリであり、RAM(Random Access Memory)で構成される。この第2メモリは、第2CPUが使用する。第2CPU2900は、暗号化部2100、復号化部2200等の処理を実行する機能を有する。

- [0034] ここで、耐タンパ実行部2000と、その外部の仮想マシン1300等との通信は、盜聴等できないように保護されているものとする。また、耐タンパ実行部2000内の第2メモリや各記憶部は、耐タンパ実行部2000の外部からはアクセスできないものとする。ただし、耐タンパ実行部2000からは、耐タンパ実行部2000の外部の第1メモリや記憶部にはアクセスできるものとする。
- [0035] また、第1CPU1900と第2CPU2900とは独立して動作できるものとする。ここで第1CPU1900と第2CPU2900とが独立して動くとは、例えば、実行装置1000が複数のCPUを備え、そのうちの1つのCPUを第2CPU2900として利用する場合が該当する。
- 尚、実行装置1000の各機能は、実行装置1000のメモリ又はハードディスクに格納されているプログラムをCPUが実行することにより実現される。
- [0036] <データ>
- 以下、本実行装置1000で用いる主なデータについて、図2から図7を用いて説明する。
- 図2は、クラスファイルの構成例を示す図である。アプリケーション1200は、複数のクラスファイルで構成されており、このクラスファイルを、仮想マシン1300が解釈実行していく。
- [0037] クラスファイル1210は、複数の要素で構成されているが、ここでは本実施形態において、特に関係のあるmagic_number1211、minor_version1212、major_version1213、fields_count1214、fields[fields_count]1215、methods_count1216、methods[methods_count]1217およびattributes[attributes_count]1218のみを説明し、他の要素の詳細な説明は省略する。
- [0038] magic_number1211は、Java(登録商標)のクラスファイルであることを示すものである。Java(登録商標)クラスファイルは必ず「CAFEBABE」という固定値から始まる。この値が「CAFEBABE」でないクラスファイルは、ベリファイ処理を通過できないことになる。
- minor_version1212およびmajor_version1213は、このクラスファイルを作成したバイトコードコンパイラのバージョンを表す。このバージョンを仮想マシン1300が

サポートしている場合のみ、仮想マシンによって実行される。尚、minor_version1212はこのクラスファイルを作成したコンパイラのマイナーバージョンを示し、major_version1213は、メジャー・バージョンを示す。

[0039] fields_count1214は、fieldsの個数であり、fields[fields_count]1215は、このクラスがもつフィールドについての情報の配列である。

また、methods_count1216は、methodsの個数であり、methods[methods_count]1217は、このクラスが持つメソッドについての情報の配列である。

[0040] attributes[attributes_count]1218は、このクラスが持つ属性情報の配列である。この属性情報は、ユーザが新たに追加することが可能であり、本実施形態では、暗号強度についての情報を追加しているものとする。

次に、図3はクラスブロックの構成例を示した図である。

クラスブロック1701とは、クラスファイル1210(図2参照)を、仮想マシン1300の内部形式に変換したデータをいうものとする。具体的には、クラスファイル1210が、クラスローダ1320によってメモリ上に展開されたものである。

[0041] クラスブロック1701は、クラスヘッダ1710、メソッド1800を含む。

クラスヘッダ1710とは、クラスの分割情報などを保持しているデータであり、本発明で新たに追加されたデータである。このクラスヘッダの詳細は、図4および図5を用いて、後で説明する。

また、メソッド1800は、メソッドヘッダ1810とコード1850とを含む。

[0042] メソッド1800は、methods[methods_count]1217を、仮想マシン1300の内部形式に変換したものであり、methods_counts1216の数だけ存在する。

メソッドヘッダ1810は、メソッドの暗号化についての情報などを保持しているデータであり、本発明で新たに追加されたデータである。このメソッドヘッダの詳細は、図4および図5を用いて、後で説明する。

[0043] また、コード1850は、このメソッドの機能を実行するための一連の命令コードであり、ほとんどのメソッドに含まれる。具体的には、バイトコード、ネイティブコード、またはネイティブコードに最適化処理を施した最適化ネイティブコードの何れかである。

尚、クラスファイルのメソッド以外の情報も、分割され内部形式に変換されてクラスブ

ロック1701に含まれているものとし、同様に、メソッドにもコード以外の情報が内部形式に変換されて含まれているものとする(図示していない。)。

[0044] 次に、図4を用いて、クラスヘッダ1710およびメソッドヘッダ1810を説明する。

図4(a)は、クラスヘッダの構成例を示す図であり、図4(b)は、メソッドヘッダの構成例を示す図である。

クラスヘッダ1710は、暗号強度情報1720と分割情報1730とで構成される。

[0045] 暗号強度情報1720は、このクラスを暗号化する場合の暗号の強度を表したものであり、このクラスの秘匿性の強さを表したものといえる。

分割情報1730は、このクラスをどのように分割したかを表す情報であり、この分割部分が暗号化の単位となる。分割は、参照するタイミングが同じものや、重要度が同程度のものを1つの分割部分とするなどして、暗号化等の処理を軽くするような分割とすることも可能である。本実施形態では、クラスのメソッドは、1メソッドで1分割部分とする。

[0046] このクラスヘッダ1710を作成する基となる情報は、クラスが有している属性情報(図2の1218参照)に設定されているものとする。

メソッドヘッダ1810は、暗号強度情報1811、JITヘッダ1812、最適化ヘッダ1813およびメソッド情報1814で構成される。

まず、暗号強度情報1811は、このメソッドを暗号化する場合の暗号の強度を表したものであり、クラスヘッダの暗号強度情報1720と同じものである。本実施形態では、メソッドの暗号強度情報1811がクラスの暗号強度情報1720よりも強度が強い場合に、メソッドの暗号強度情報1811を採用し、弱い場合には、クラスの暗号強度情報1720を採用するものとする。

[0047] JITヘッダ1812は、コード1850に対してJITコンパイルを行うか否か、またさらに、コード1850がJITコンパイル済かどうかを示すの情報を保持している。

最適化ヘッダ1813は、コード1850に対して最適化を行うか否か、また、現在の最適化のレベル、目標となる最適化レベルなどが記されている。さらに、最適化ヘッダ1813には、実行速度やコードサイズなど、実施する最適化の種類または割合も記されている。

[0048] メソッド情報1814は、methods313に含まれるバイトコード以外の情報を、仮想マシン1300の内部形式に変換したデータである。

このメソッドヘッダ1810を作成する基となる情報は、クラスと同様に、メソッドが有している属性情報に設定されているものとする。

図5(a)は、暗号強度情報(1720、1811)の内容例を表す図であり、図5(b)は、分割情報1730の構成例および内容例を表す図である。

[0049] 図5(a)において、例えば、暗号強度情報1720は「1」が指定してある。この「1」に基づいて、暗号化部2100(図1参照)が暗号方式を決定することになる。本実施形態では、この値が「0」の場合は、暗号化が不要である旨を表し、他に「1」、「2」、「3」のいずれかが設定でき、この値が大きいほど秘匿性が高い、すなわち、強い強度で暗号化することを要求する旨であるとする。

[0050] 図5(b)は、分割情報1730であり、項目1731とアドレス1732とで構成される。

項目1731は、クラスファイルの項目であり、アドレス1732は、第1メモリに配置されたアドレスである。

例えば、項目1731「magic_number、minor_version、major_version」の3項目が1つの分割単位となり、アドレス1732「0x0430012d」に配置されており、「fields[0]～fields[10]」が、アドレス1732「0x5d8cb321」に配置されている。

[0051] 次に、図6は、暗号化方式情報の構成例および内容例を表す図である。

この暗号化方式情報2301は、予め決められているものとし、暗号化方式情報記憶部2300に記憶されている。

暗号化方式情報2301は、鍵ID2310、暗号鍵2320、復号鍵2330および暗号アルゴリズム2340で構成される。

[0052] 鍵ID2310は、鍵の識別番号を表す。

暗号鍵2320は、暗号化の際に使用される鍵であり、復号鍵2330は、復号の際に使用される鍵である。

暗号アルゴリズム2340は、暗号化のアルゴリズムを表している。本実施形態では、「A」「B」「C」には、予め、ある暗号アルゴリズムが割り当てられているものとする。例えば、DES(Data Encryption Standard)、RSA(Rivest Shamir Adleman)などである。

[0053] 例では、鍵ID2310が「1」の暗号化方式の暗号鍵2320は「fd456」、復号鍵2330は「4fgaa」、暗号アルゴリズム2340は「A」である。

図7は、暗号鍵情報の構成例および内容例を表す図である。

この暗号鍵情報2401は、暗号鍵情報記憶部2400に記憶されている。

暗号鍵情報2401は、アドレス1732と鍵ID2310とで構成される。

[0054] アドレス1732は、暗号化されたデータが入っている第1メモリのアドレスである。このアドレス1732は、クラスの分割部分が配置されているアドレスであり、分割情報1730(図5参照)のアドレスと同じものである。

このアドレス1732の指す先のデータは、暗号化されているデータの場合もあれば、復号化されているデータの場合もある。

[0055] 鍵ID2310は、暗号化方式情報1301の鍵ID2310と同じものである。この鍵ID2310で、暗号鍵情報2401と暗号化方式情報2301とがリンクしていることになる。

例えば、アドレス「0x06530012」のデータは、鍵ID「4」の暗号化方式で暗号化または復号化されており、その暗号化方式は、暗号鍵2320「135fd」、復号鍵2330「2ljiol」、暗号アルゴリズム2340「C」である(図6参照)。

[0056] <動作>

以下、本発明に係る実行装置1000の動作について図8～図15を用いて説明する。

本実施形態では、実行装置の電源を入れた場合、予め定められているアプリケーションが実行されるものとする。

図8は、実行装置1000の処理を表すフローチャートである。

[0057] まず、ユーザが実行装置1000の電源を投入する(ステップS100)。

通電した第1CPU1900は、OS1400を起動し(ステップS110)、起動されたOS1400は、仮想マシン1300を起動する(ステップS120)。

起動された仮想マシン1300は、アプリ取得プログラム1100を起動する(ステップS130)。

[0058] 起動されたアプリ取得プログラム1100は、アプリケーション1200を実行装置1000の外部から読み込み、アプリケーションの実行を開始する(ステップS140)。

仮想マシン1300は、適時、アプリケーションのネイティブコードなどを暗号化したり、復号化したりして(ステップS170)、仮想マシン1300内のアプリケーションを保護しながら、実行する。

- [0059] 暗号化の際には、暗号化するデータのアドレスと、暗号強度情報(1720、1811)を渡し、復号化の際には、復号するデータのアドレスを渡す。

アプリケーションの実行が終了すると(ステップS150:YES)、終了処理を行い(ステップS160、ステップS180)、実行装置における処理を終了する。

ここでは、アプリケーション実行中の仮想マシン1300の主な処理である、クラスロード処理(ステップS200)とベリファイ処理(ステップS300)、メソッド実行処理(ステップS400)について、以下で説明する。

- [0060] これらの処理の説明後、暗号化処理と復号化処理(ステップS170)について説明する。

<クラスロード処理>

図9は、クラスロード処理を表すフローチャート、および第1メモリの内容を表す図である。フローチャートの左側に示している図が、第1メモリの内容である(9000～9003)。このメモリの内容は、本発明に関するもののみを時系列で表している(図10～図15において同様。)。

- [0061] このクラスロード処理は、クラスローダ1320が行う処理であり、クラスファイルを内部形式にして、メモリ上に配置する処理を行う。

本実施形態では、この際、分割して、分割部分ごとに暗号化してメモリ上に配置する。尚、通常、クラスロードの処理中に行われるベリファイの処理は、説明の便宜上省くものとし、ベリファイの処理は図10を用いて説明する。

- [0062] インタプリタ1310より、クラス名を渡されロードを指示されたクラスローダ1320は、渡されたクラス名に該当するクラスファイルを探し出し、第1メモリに読込む(ステップS210、第1メモリ9000、9001)。この際、クラスファイルが暗号化されている場合は、復号化する処理を行う。

読み出されたクラスファイルを基に、クラスヘッダ1710を作成する(ステップS220、第1メモリ9002)。

[0063] 本実施形態では、暗号強度情報1720は、クラスのattributes[attributes_count]1218に、新たな属性として入っている。

また、クラスを分割して、分割情報1730を作成する(ステップS220)。本実施形態では、図5(b)のように分割するものとする。

次に、クラスブロック1701を作成する(ステップS230、第1メモリ9002)。具体的には、クラスブロックを構成する各要素を作成することになる。例えば、メソッドヘッダ1810とコード1850とを作成し、メソッド1800をメソッドの数分作成する。この際、分割情報1730に、メソッドヘッダ1810とコード1850を、それぞれ1分割部分として追加する。その他、分割情報1730のアドレス1732が変わる場合は、書き換える。

[0064] クラスブロックを作成したら、クラスブロックを暗号化する処理を行う(ステップS240)。暗号化は、耐タンパ実行部2000に依頼して行なう(ステップS240:矢印)。この暗号化の処理単位は、クラスヘッダ1710の分割情報1730を基に行い、最後にクラスヘッダ1710の暗号化を行う。クラスヘッダの分割情報1730のアドレスは、暗号化された各分割部分を指していることになる。すなわち、暗号化、復号化を行う都度、分割情報1730の該当アドレスは書き換えられることになる(以下、同様)。

[0065] また、暗号化されたクラスヘッダのアドレスは、インタプリタ1310が管理している。

暗号化クラスブロックを作成したら、第1メモリ上のクラスヘッダ、クラスファイルを削除する(ステップS250、第1メモリ9003)。

ここでの削除とは、例えば、記録しているデータを管理しているインデックスのみを削除するような削除ではなく、記録しているデータを特定の値、例えば「0x00」で、全て上書きするような削除を意味する(以下、同様)。

[0066] この時点で、第1メモリ上には、目的とするクラスファイルはロードされているが、平文のクラスではなく、暗号化されたクラスのみ存在していることになり、盗聴、改竄から保護されている。

<ベリファイ処理>

図10は、ベリファイ処理を表すフローチャート、および第1メモリの内容を表す図である。フローチャートの左側に示している図が、第1メモリの内容である(9010~9012)。

[0067] このベリファイ処理は、ベリファイヤ1330が行う処理である。このベリファイ処理は、クラスファイルのロード時や、リンク時など適時に呼び出され、呼出し時に応じた内容のベリファイ処理を行う。

本実施形態では、暗号化クラスブロックに対して、何らかのベリファイ処理を行う場合について説明する。

[0068] インタプリタ1310から、対象となる暗号化クラスファイル(第1メモリ9010)を渡されたベリファイヤ1330は、その暗号化クラスブロックを復号化する処理を行う(ステップS310、第1メモリ9011)。この復号化では、ベリファイ処理に必要な分割部分のみを復号する。例えば、マジックナンバーが「CAFEBABE」になっているかを確認する場合は、クラスヘッダ1710と、分割情報1730の最初の分割部分を復号する。

[0069] 復号化クラスブロックをもとに、ベリファイの処理を行う(ステップS320)。この際、他のクラスが必要となる場合は、そのクラスブロックを復号したり、ロードしたり等して、ベリファイ処理を行う。

ベリファイ処理によって、分割情報1730に変更があれば、更新する(ステップS330)。

[0070] その後、クラスブロックの暗号化を行う(ステップS340)。この場合は、クラスブロックのうち、復号化した分割部分のみを再暗号化することになる。

復号化クラスブロックを削除する(ステップS350、第1メモリ9012)。

<メソッド実行処理>

図11は、メソッド実行処理を表すフローチャート、および第1メモリの内容を表す図である。フローチャートの左側に示している図が、第1メモリの内容である(9020～9023)。

[0071] このメソッド実行処理は、インタプリタ1310が行う処理である。

本実施形態では、実行するメソッドが属するクラスは、ロードされて暗号化クラスブロックとして存在している場合について説明する(第1メモリ9020)。

また、JITコンパイルの処理は、クラスのロード時に行われることが多いが、本実施形態では、メソッドの実行前に行われる場合を説明する。

[0072] インタプリタ1310は、実行するメソッドの属する暗号化クラスブロックの中の、該当メ

ソッド1800のメソッドヘッダ1810を復号する(ステップS410、第1メモリ9021)。具体的には、クラスヘッダ1710を復号し、分割情報1730から該当メソッドのクラスヘッダのアドレスを取得して、メソッドヘッダ1810を復号する。

メソッドヘッダ1810のJITヘッダ1812を参照し、このメソッドをJITコンパイルする場合(ステップS420:YES)は、JITコンパイル処理を行う(ステップS500)。

- [0073] また、メソッドヘッダ1810の最適化ヘッダ1813を参照し、このメソッドを最適化する場合(ステップS430:YES)は、最適化処理を行う(ステップS600)。

その後、分割情報1730から該当メソッドのコードのアドレスを取得して、コード1850を復号化し(ステップS440、第1メモリ9022)、実行する(ステップS450)。この際、他のメソッドを実行する場合は、そのメソッドを復号するなどの処理を行う。

- [0074] メソッドの実行が終了したら、復号化コードと復号化メソッドヘッダを削除する(ステップS460、第1メモリ9023)。

<JITコンパイル処理>

図12は、JITコンパイル処理を表すフローチャート、および第1メモリの内容を表す図である。フローチャートの左側に示している図が、第1メモリの内容である(9030～9032)。

- [0075] このJITコンパイル処理は、主に、JITコンパイラ1340が行う処理である。

インタプリタ1310は、JITコンパイラ1340にJITコンパイルするコード1850を渡し、コンパイルを指示する(第1メモリ9030)。ここでのコードは、バイトコードである。

指示されたJITコンパイラ1340は、渡された暗号化コードを復号する(ステップS510、第1メモリ9031)。

- [0076] その後、JITコンパイラ1340は、復号化コードから、ネイティブコードを作成する(ステップS520、第1メモリ9031)。

インタプリタ1310は、作成されたネイティブコードを、基となったコード1850と同じメソッド1800内に、暗号化して追加する(ステップS530)。

復号化コードとネイティブコードを削除する(ステップS540、第1メモリ9032)。

- [0077] <最適化処理>

図13は、最適化処理を表すフローチャート、および第1メモリの内容を表す図であ

る。フローチャートの左側に示している図が、第1メモリの内容である(9040～9043)。

◦

この最適化処理は、オプティマイザ1350が行う処理である。

- [0078] インタプリタ1310は、オプティマイザ1350に最適化するコード1850を渡し、最適化を指示する。このコード1850が属するクラスは既にロードされ、暗号化クラスブロックとして第1メモリにあるものとする(第1メモリ9040)。また、ここでのコードは、バイトコードまたはネイティブコードである。

指示されたオプティマイザ1350は、渡された暗号化コードを復号する(ステップS610、第1メモリ9041)。

- [0079] その後、オプティマイザ1350は、復号化コードを基に、最適化されたコードを作成する(ステップS620、第1メモリ9042)。最適化に際し、他に必要な分割部分があれば、復号化等して処理を行う。

インタプリタ1310は、作成されたコードを暗号化し(ステップS630)、基となったコード1850と置き換える(ステップS635)。

- [0080] 最適化の対象となった復号化コードと、作成した最適化された復号化コードとを削除する(ステップS650、第1メモリ9043)。

<暗号化処理>

図14は、暗号化処理を表すフローチャート、および第1メモリ、第2メモリの内容を表す図である。フローチャートの左側に示している図が、第1メモリ(9050～9053)および第2メモリ(9060～9063)の内容である。

- [0081] この暗号化処理は、耐タンパ実行部2000の暗号化部2100が行う処理である。

仮想マシン1300は、暗号化部2100に、暗号化するデータのアドレスと暗号強度情報(1720, 1811)とを渡し、暗号化を指示する(第1メモリ9050、第2メモリ9060)。

◦

指示された暗号化部2100は、渡されたアドレスのデータを、第1メモリから第2メモリへと読み込み(ステップS710、第2メモリ9061)、第1メモリのデータを削除する(ステップS720、第1メモリ9051)。

- [0082] 暗号化部2100は、受け取った暗号強度情報を基に、暗号化方式(図6、図7参照)

を決定する(ステップS730)。例えば、暗号強度情報が強い暗号化を求めるものであれば、暗号アルゴリズム2340の内、強い暗号アルゴリズムを選択し、同じ暗号鍵2320が連続しないように、暗号化方法を決定する。

決定した暗号化方式である暗号アルゴリズム2340の、暗号鍵2320で、データを暗号化する(ステップS750、第2メモリ9062、第1メモリ9052)。

- [0083] データを暗号化したら、暗号化したデータを第1メモリに書き込み(ステップS760、第1メモリ9053、第2メモリ9063)、書込んだアドレスと、暗号化した暗号の鍵ID2310とを対応付けて、暗号鍵情報2401に追加する(ステップS770)。

尚、第2メモリ2700は、耐タンパ実行部2000の中にあるため、データの削除は要しない。

- [0084] <復号化処理>

図15は、復号化処理を表すフローチャート、および第1メモリ、第2メモリの内容を表す図である。フローチャートの左側に示している図が、第1メモリ(9070～9073)および第2メモリ(9080～9083)の内容である。

この暗号化処理は、耐タンパ実行部2000の暗号化部2100が行う処理である。

- [0085] 仮想マシン1300は、復号化部2200に、復号化する暗号化データのアドレスを渡し、復号化を指示する(第1メモリ9070、第2メモリ9080)。

指示された復号化部2200は、渡されたアドレスの暗号化データを、第1メモリから第2メモリへと読み込み(ステップS810、第2メモリ9081、第1メモリ9071)、第1メモリの暗号化データを削除する(第1メモリ9072)。

- [0086] 復号化部2200は、受け取ったアドレスと同じアドレスがないか、暗号鍵情報2401のアドレス1732をサーチする。同じアドレスがあれば、そのアドレス1732に対応する鍵ID2310を、暗号化されたときの鍵IDであると特定する(ステップS820)。

次に、暗号化方式情報2301から、特定した鍵ID2310に対応する復号鍵2330と暗号アルゴリズム2340とを取得し(ステップS830)、データを復号化する(ステップS840、第2メモリ9082、第1メモリ9072)。

- [0087] データを復号化したら、復号化したデータを第1メモリに書き込む(ステップS850、第1メモリ9073、第2メモリ9083)。

<実施形態2>

本実施形態は、実施形態1における耐タンパ実行部2000が行っている暗号化処理、復号化処理の替わりに、OS1400が提供するメモリ管理機能を使うものである。具体的には、JITコンパイルされたネイティブコードや中間コードがロードされているメモリ領域である第1メモリ1700(図1参照)の、読み込み可能属性を変更させることで実現する。

- [0088] 本実施形態の機能ブロック図は、実施形態1の機能ブロック図である図1から、耐タンパ実行部2000を削除し、第1メモリ1700の換わりに、図16で説明するメモリ3700を用いたものである。

図1および図16を用いて、動作を説明する。

図16(a)、(b)は、JITコンパイルされたコードを記憶しているメモリ領域に対し、メモリ上の読み込み可能属性と読み込み不可能属性を設定した際のメモリの状態を表す図である。

- [0089] 図16(a)において、メモリ3700は、メモリ3710とメモリ3720とに分かれており、双方のメモリに読み込み可能属性が設定されている。

この時、仮想マシン1300や、デバッガなどの攻撃プログラムは、JITコンパイルされたコードを読み出すことが可能である。

図16(b)において、メモリ3711は、読み込み不可能属性が設定されており、このとき、仮想マシン1300や、デバッガなどの攻撃プログラムはJITコンパイルされたコードを読み出すことはできない。

- [0090] 仮想マシン1300は、通常はJITコンパイルされたコードを記憶しているメモリ領域を読み込み不可能に設定しておき、JITコンパイルされたコードを実行する場合や、中間データを必要とする場合に、必要に応じて読み込み可能に設定する。

このように構成することにより、メモリ3700が記憶するデータの内、盗まれる可能性のあるコードは、第1CPU1900により実行中のコードのみに限定されるため、改竄等の被害を最小限に抑えることができる。

- [0091] さらに、これにより、暗号化、復号化の処理を行う必要がないため、実行速度を落とすことなく、JITコンパイルされたプログラムを保護することができるという利点がある。

メモリの読み込み可能属性を変更できるものとして、次のようなものがある。

例えば、Linux OSでは、メモリの読み込み可能、書き込み可能、実行可能などの属性を変更させるシステムコールmmap()を備えている。これを用いることで、例えば、JITコンパイルされたコードがロードされているメモリ領域に対して、メモリ領域の読み取許可を指定するPROT_READのフラグを無効にすることで、デバッガなどのツールからJITコンパイルされたコードが盗聴されることを防ぐことができ、暗号化と同様の効果が得られる。

[0092] 仮想マシン1300がJITコンパイルされたコードを必要とする際には、前記JITコンパイルされたコードがロードされているメモリ領域に対し、PROT_READフラグを有効にすることで、再度アクセスすることができるようになる。

また、このメモリの読み込み可能属性を変更させる機能は、ハードウェアで実現されてもよい。

[0093] <考察>

以下、従来技術を説明し、本発明との違いを示す。

図17に示すように、従来のデータ保護機能を備えた計算機は、主に暗号化されたプログラム201と復号化手段202と仮想機械203とCPU204から構成されている。

復号化手段202と仮想機械203は、CPU204で実行可能なネイティブコードで記述されており、暗号化プログラム201は、仮想機械203が解釈実行可能な命令セットで記述されたプログラムを暗号化したものである。

[0094] ソースコードからソフトウェアの実行イメージ(いわゆる内部形式でメモリ上に展開されたものを生成する時に、仮想機械の独自の命令セットを生成し、ソースコードを前記命令セットへとコンパイル、暗号化し、前記仮想機械203と復号化手段202をリンクする。

ここで使用する独自の命令セットは、公開されない情報であるため、攻撃者は復号された暗号化プログラム201を見てもその内容を理解できることになり、結果として、中間コードレベルでの、攻撃者による盗聴、改竄を防ぐことが出来る可能性が高くなる。

[0095] しかし、本発明に係る実行装置は、中間コードを実行するよりも実行速度が速いネ

イティブコードを作成した場合においても、アクセス可能なメモリ上に置かれたネイティブコードを盗聴、改竄から守る事が出来る。

<補足>

以上、本発明に係る実行装置について実施形態に基づいて説明したが、この実行装置を部分的に変形することもでき、本発明は上述の実施形態に限られないことは勿論である。即ち、

(1) 実施形態では、アプリケーションのクラスを分割し、メソッドの単位で暗号化することとしているが、メソッドのコードを分割することとしてもよい。

- [0096] 分割部分は、分岐命令を含まない一連の命令群とすることで、実行時に平文となるネイティブコードの量を、より少なくすることが可能である。

この場合、例えば、メソッドヘッダ内に、クラスヘッダ内の分割情報と同様の分割情報を持ち、暗号化等を行う。

また、1命令ずつに分割することとしてもよい。この場合は、特定の命令のみを暗号化したい場合に特に有効である。

(2) 実施形態では、クラスヘッダの暗号強度情報1720で、暗号化を行わない旨の指定がない限り、暗号化を行うこととしているが、暗号化処理、復号化処理は、任意の組み合わせで省略してもよい。

- [0097] 例えば、クラスロード処理(図9参照)とベリファイ処理(図10参照)とを連続して行う場合には、それらの処理の間で、クラスブロックの暗号化、復号化を省略するなどである。

この場合は、暗号強度情報1720で暗号化を行わない旨の指定をするほかに、予め暗号化、復号化を行わない場合を決めておくことで、暗号化、復号化を省略することができる。

- [0098] 暗号化、復号化を省略した場合は、暗号化、復号化を行う回数が削減されるため、アプリケーションの実行速度低下を低減することができるという利点がある。また、使用メモリの削減を実現することもできるようになる。

(3) 実施形態では、実行装置1000の第1CPU1900と、耐タンパ実行部2000の第2CPU2900とは、物理的に異なるCPUであるとしているが、物理的に1つのCPUが

、動作モードを切り替えるなどの方法で仮想的に2つのCPUのように振舞うこととしてもよい。また、マルチコアCPUのように、1つのCPUパッケージのなかに複数のCPUコアを持つCPUでは、そのうちの特定のコアを第2CPUとして動作させてもよい。

- [0099] また、実施形態では、実行装置1000と耐タンパ実行部2000とにそれぞれ、RAMおよびROMを持つこととしているが、1つのRAMを仮想的に2つのRAMとして扱ってもよい。同様に、1つのROMを仮想的に2つのROMとして扱ってもよい。さらに、耐タンパ実行部2000内のROMは第2CPU2900に混載されていてもよい。
- (4) 実施形態では、暗号強度情報をクラスファイルの拡張属性に埋め込んだが、これ以外の方法を用いて暗号強度情報を指定することとしてもよい。
- [0100] 例えば、暗号強度情報を記録したXMLファイルをアプリケーションと同時にアプリ取得プログラム1100が取得し、仮想マシン1300へ渡すことにより、仮想マシン1300は、クラスなどの暗号強度情報を知ることが可能になる。
- (5) 実施形態では、実行装置1000で実行するアプリケーションは、アプリ取得プログラム1100が、本装置外のアプリケーションファイルからダウンロードしたものであるとしているが、インターネット上にあるサーバから、ダウンロードすることとしてもよい。
- [0101] この場合、アプリ取得プログラム1100は、TLS(Transport Layer Security)、HTTP(Hypertext Transfer Protocol)等のプロトコルに従いJava(登録商標)アプリケーションをダウンロードする機能を有するプログラムとなる。
- ここで、TLSは暗号化により通信時のデータの盗聴、改竄を防ぐデータ転送方式である(RFC2246参照)。また、HTTPは、インターネット上のデータ通信で一般的に用いられているデータ転送方式である(RFC2616参照)。
- [0102] 尚、RFC(Request For Comments)とは、インターネット上の技術を規格化するIETF(Internet Engineering Task Force)の公式文書であり、プロトコルなど様々な技術の使用がまとめられているものである。
- また、実行装置1000で実行するアプリケーションは、デジタル放送のデータ放送として、MPEG(Moving Picture Coding Experts Group)2トランスポートストリーム内に埋め込まれたJava(登録商標)アプリケーションであってもよい。
- [0103] この場合、アプリ取得プログラム1100は、トランスポートストリーム内に埋め込まれた

Java(登録商標)アプリケーションを実行装置1000内に読み出すプログラムとなる。

MPEG2トランスポートストリームにJava(登録商標)プログラムを埋め込む方法としては、例えば、DSMCC方式がある。DSMCC方式とは、MPEG2トランスポートストリームのパケットの中に、コンピュータで使用されているディレクトリやファイルで構成されるファイルシステムをエンコードする方法である(MPEG規格書ISO／IEC138181－1、MPEG規格書ISO／IEC138181－6参照)。

- [0104] またさらに、実行装置1000で実行するアプリケーションは、SDカード(Secure Digital memory card)、CD—ROM(Compact Disk Read Only Memory)、DVD(Digital Versatile Disk)、Blu—RayDisc等に記録されたJava(登録商標)アプリケーションであってもよい。

この場合、アプリ取得プログラム1100は、これらの記録媒体からアプリケーションを読み出すプログラムとなる。

- [0105] また、実行装置1000で実行するアプリケーションは、実行装置1000内にあるROMなどに記録されたJava(登録商標)アプリケーションであってもよい。

この場合、アプリ取得プログラム1100は、ROMから作業メモリに、Java(登録商標)アプリケーションを読み出すプログラムとなる。

(6) 本実施形態では、アプリ取得プログラム1100などはJava(登録商標)言語で記述されたJava(登録商標)プログラムとしているが、同等の機能を有する、ネイティブ言語で記述されたプログラムや、ハードウェアで実現されていてもよい。

- [0106] また、仮想マシンで実行するアプリケーションは、Java(登録商標)言語で記述されたものに限らず、C++などの他のオブジェクト指向言語で記述されたものであってもよく、C言語など、他の言語で記述されたものでもよい。

(7) 実施形態の耐タンパ実行部2000は、例えば、ARM社のTrustZone(登録商標)技術を使うことで実現することができる。

- [0107] TrustZone(登録商標)技術では、RAMやROMなどのハードウェア資源の一部をセキュアドメインと呼ばれる仮想的に実行環境に割り当てることができる。セキュアドメインに割り当てられたRAMやROMは、セキュアドメインで動作するプログラムのみが利用可能であり、セキュアドメイン以外で動作するプログラムからは一切利用するこ

とができない。

- [0108] 従来のCPUでは、アプリケーションが動作する通常モードとOSなどが動作する特権モードの2種類のモードを持ち、通常モードで動作するプログラムからは特権モードで動作するプログラムを改ざんすることができないようになっている。

TrustZone(登録商標)技術では、さらに、モニターモードと呼ばれる特殊なモードが新しく用意される。モニターモードへは、CPUが用意する特殊な命令を実行することで遷移することができる。モニターモードでCPUが動作するときには、S-bitと呼ばれるセキュリティ情報がRAMやROMなどの周辺ハードに通知される。TrustZone(登録商標)技術に対応したRAMやROMは、S-bitが通知された場合に限り、セキュアドメインに割り当てられた領域へのデータの読み書きを許可するように構成されている。またセキュアドメインに割り当てられていない領域へのデータへの読み書きはS-bitが通知されているか否かに関わらず読み書きを許可する。このように、安全な実行部はセキュアドメインにより実現することができる。

- [0109] また、Intel社のLaGrande技術でも、通常のアプリケーションやOSが動作するドメインと保護が必要なアプリケーションが動作するドメインを仮想的に分離するなど、TrustZone(登録商標)技術と同様の機能を提供している。このような技術を使うことで安全な実行部を実現することができる。

(8) 実施形態では、耐タンパ実行部2000は、実行装置1000の内蔵されているものとしているが、実行装置1000から着脱可能なスマートカードやICカードなどであることもしてもよい。このスマートカード、ICカードはカード内部にCPUやメモリ、セキュリティ回路を含むものである。

- [0110] 耐タンパ実行部2000は、その全体をハードウェアにより実現してもよい。この場合、第1CPUと第2CPU間のデータ通信は暗号化して行い、第三者による盗聴を防ぐことが必要となる。具体的には、両CPUを結ぶデータバス(図示していない)を介してデータを送信する際に、送信するデータを暗号化し、データを受信後に復号することで行われる。

(9) 実施形態で示した実行装置の各機能を実現させる為の各制御処理(図1等参照)をCPUに実行させる為のプログラムを、記録媒体に記録し又は各種通信路等を介

して、流通させ頒布することもできる。このような記録媒体には、ICカード、光ディスク、フレキシブルディスク、ROM、フラッシュメモリ等がある。流通、頒布されたプログラムは、機器におけるCPUで読み取り可能なメモリ等に格納されることにより利用に供され、そのCPUがそのプログラムを実行することにより実施形態で示した実行装置の各機能が実現される。

産業上の利用可能性

[0111] Java(登録商標)アプリケーションの実行環境である仮想マシンの内部で処理しているアプリケーションの、盗聴、改竄の機会を大幅に減らすことができるので、今後本格展開が予想されるJava(登録商標)アプリケーションのダウンロード配信ビジネスにおいて、コンテンツ作成者の権利を保護すること場合などに、特に有用である。

ダウンロード配信ビジネスとして、例えば、携帯電話機では、NTT DoCoMoがi-アプリと呼ばれるサービスを提供している。このサービスは、携帯電話端末がインターネット上にあるアプリケーション配信サーバからJava(登録商標)プログラムをダウンロードして、端末上で実行する。また、欧州では、DVB-MHP(Digital Video Broadcasting-Multimedia Home Platform)と呼ばれる仕様が策定され、既に仕様に準拠した運用が開始されている。DVB-MHP規格に基づくデジタル放送では、放送波に多重化されたJava(登録商標)プログラムをデジタルTVが受信し、それを実行する。

請求の範囲

- [1] 命令で構成される部分を複数個有するアプリケーションプログラムを実行する実行装置であつて、
部分からネイティブコードを特定のメモリ領域に作成し、作成したネイティブコードを暗号化して暗号コードを作成した後、前記作成したネイティブコードを削除し、作成した暗号コードを暗号コード記憶手段に記憶するネイティブコード作成手段と、
暗号コード記憶手段に記憶されている暗号コードを復号し、ネイティブコードを特定のメモリ領域に作成する復号手段と、
各部分を実行する際に、当該部分に該当する暗号コードを前記復号手段により復号し、ネイティブコードを実行した後、実行したネイティブコードを削除する実行制御手段と
を備えることを特徴とする実行装置。
- [2] 前記アプリケーションプログラムは、更に、各部分ごとに暗号化するか否かを示す秘匿性情報を含み、
前記ネイティブコード作成手段および前記実行制御手段は、前記秘匿性情報が暗号化することを示している部分に限り、当該部分を対象として処理することを特徴とする請求項1記載の実行装置。
- [3] 前記秘匿性情報は、更に、秘匿の度合いを示す情報を含み、
前記ネイティブコード作成手段は、前記秘匿性情報が暗号化することを示している場合に限り、前記秘匿性情報に基づいて決定された暗号化方式で暗号化した暗号コードを作成することを特徴とする請求項2記載の実行装置。
- [4] 前記部分は、分岐命令を含まない一連の命令群であることを特徴とする請求項1記載の実行装置。
- [5] 前記アプリケーションプログラムは、オブジェクト指向言語で作成され、
前記部分は、メソッドであることを特徴とする請求項1記載の実行装置。
- [6] 前記部分は、1つの命令である

ことを特徴とする請求項1記載の実行装置。

[7] 命令で構成される部分を複数個有するアプリケーションプログラムを実行する実行装置であって、

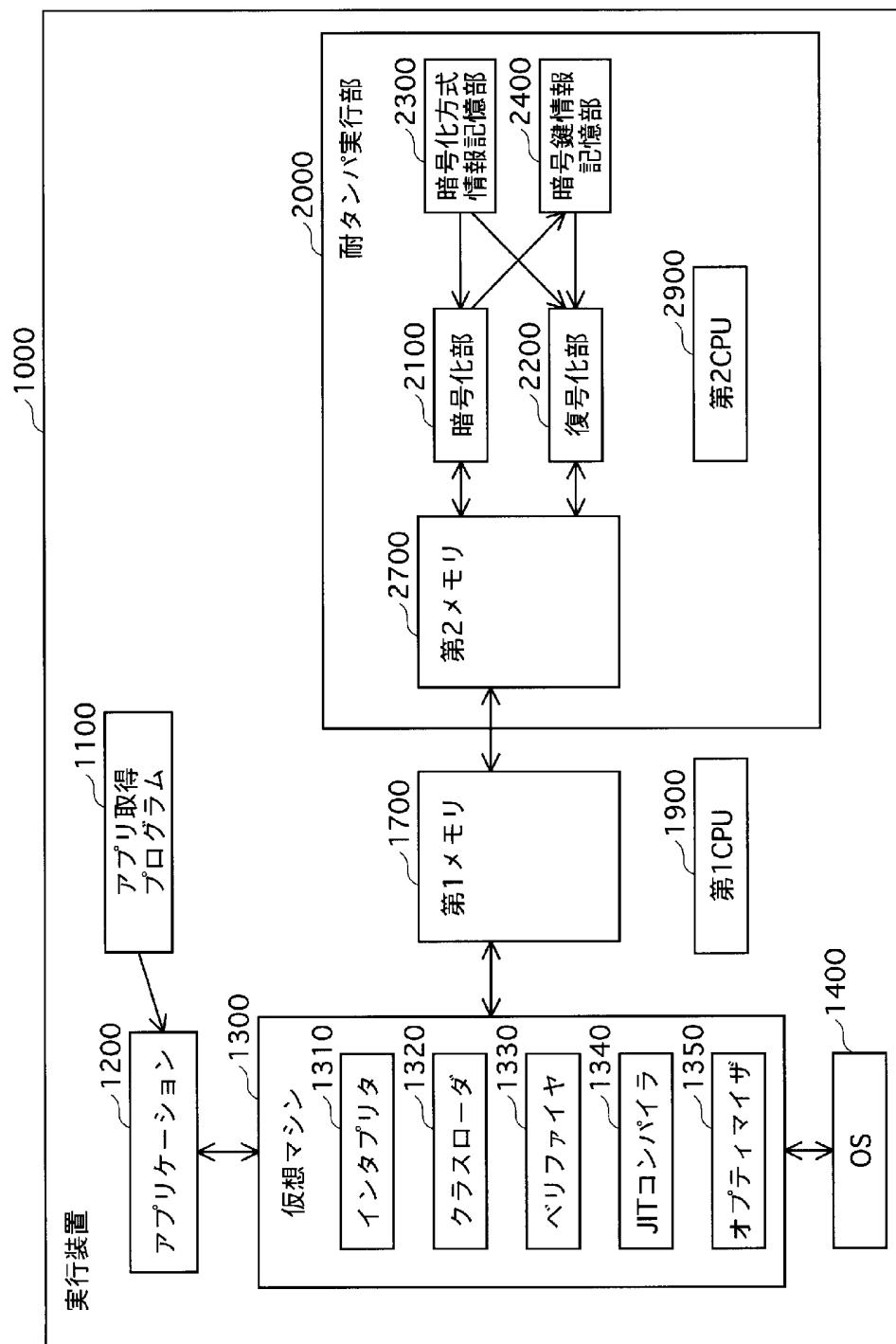
所定のメモリ領域への読み出し及び書き込みのアクセスを、許可又は禁止するよう制御するアクセス制御手段と、

前記アクセス手段により、前記所定のメモリ領域への書き込みのアクセスを許可し、部分からネイティブコードを前記所定のメモリ領域に作成し、前記アクセス手段により、前記所定のメモリ領域への書き込みのアクセスを禁止するネイティブコード作成手段と、

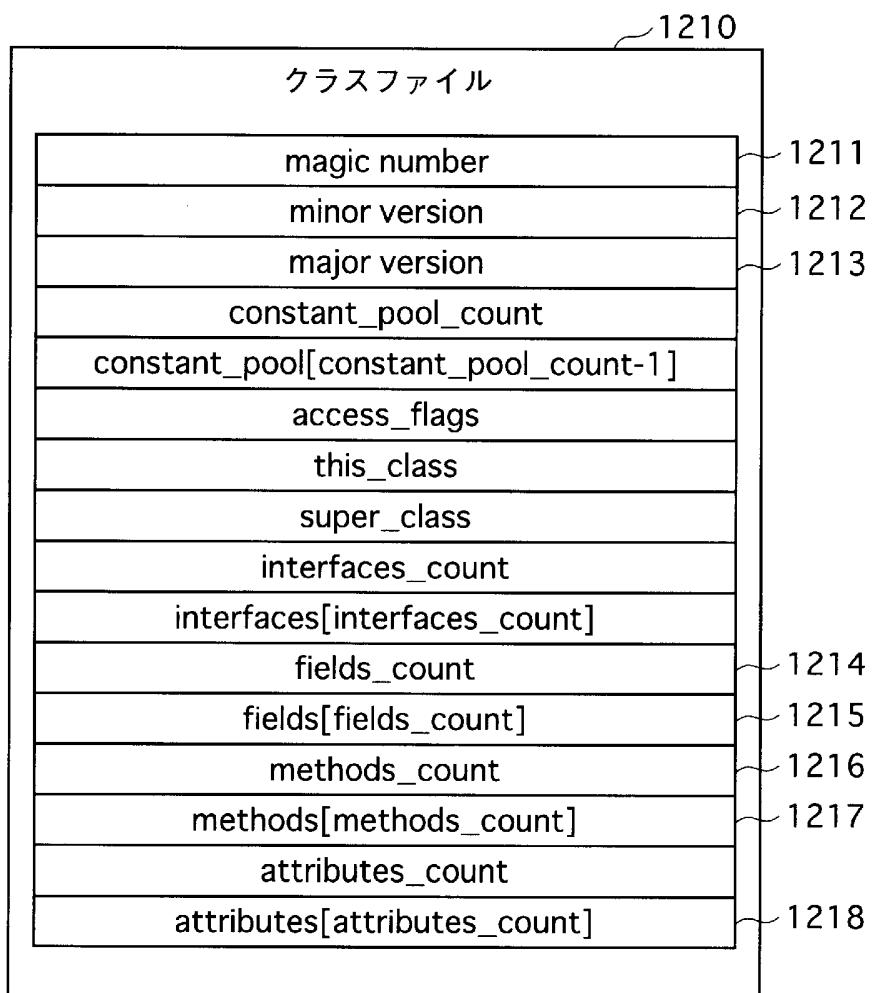
部分を実行する際に、前記アクセス手段により、前記所定のメモリ領域への読み込みのアクセスを許可し、当該部分に該当するネイティブコードを読み出した後、前記アクセス手段により、前記所定のメモリ領域への読み込みのアクセスを禁止する実行制御手段と

を備えることを特徴とする実行装置。

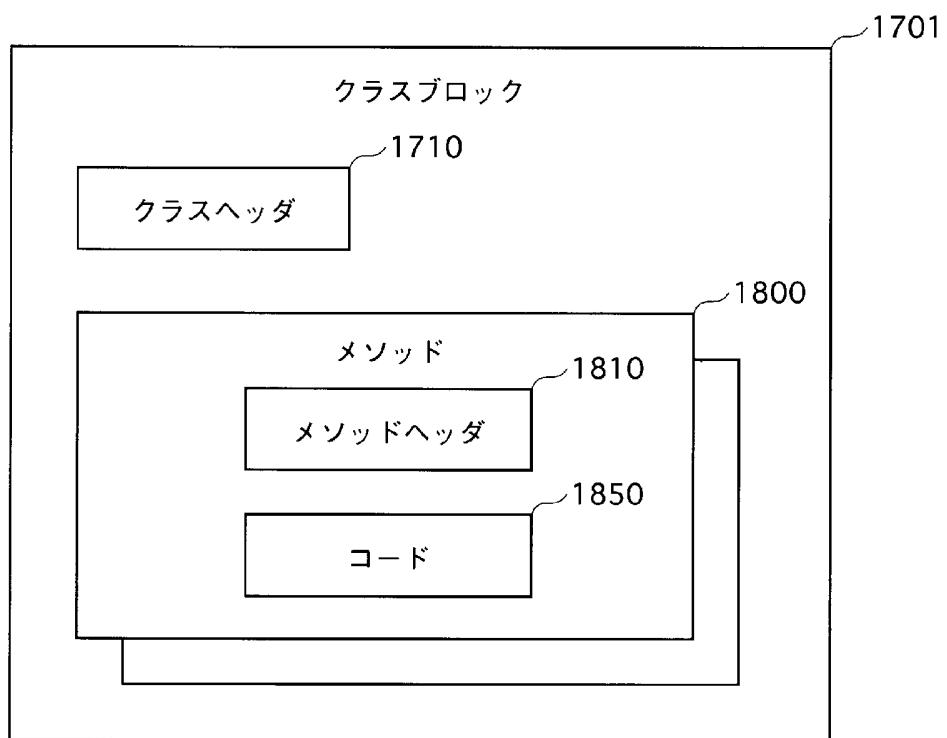
[図1]



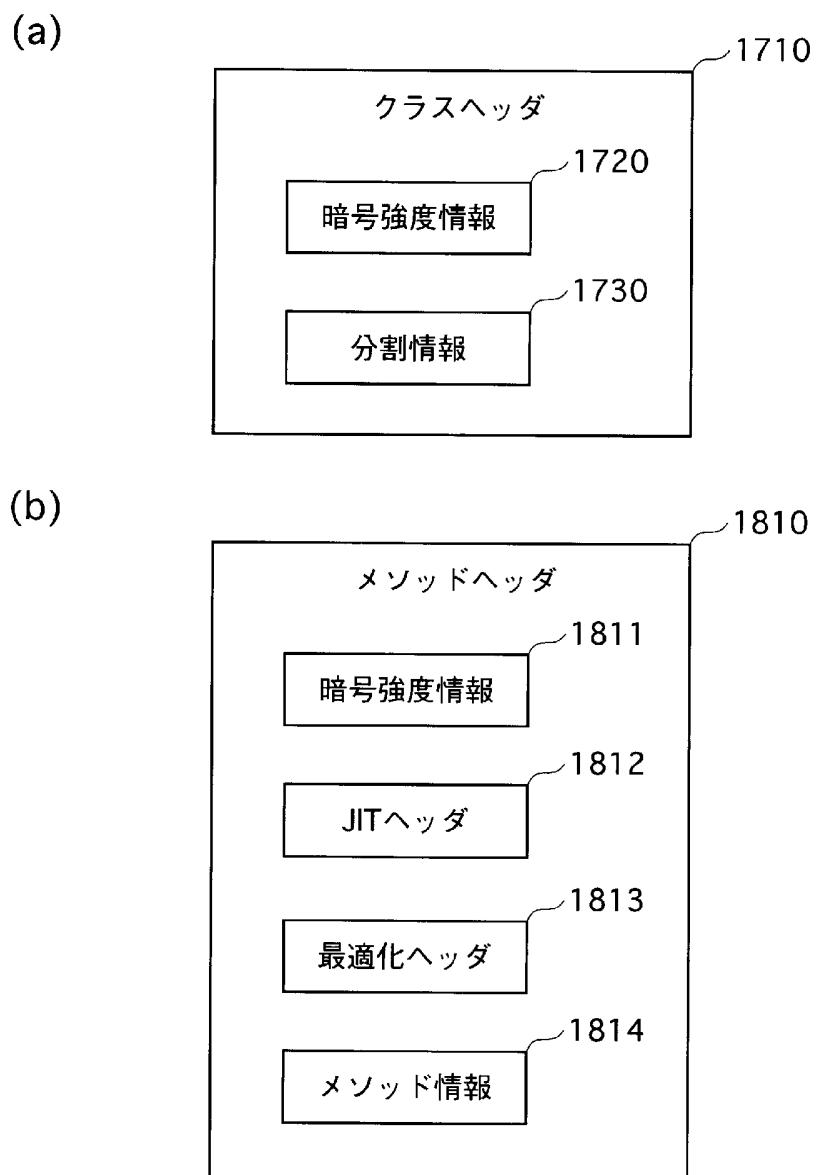
[図2]



[図3]

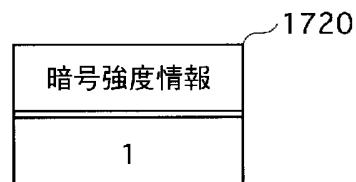


[図4]



[図5]

(a)



(b)

項目	アドレス
magic number minor version major version	0x0430012d
fields[0]-[10]	0x5d8cb321
fields[11]-	0x5e0abcd
methods[0]	0x0623bc8d
methods[1]	0x06300fcl
methods[2]	0x06530012

1731 1732 1730

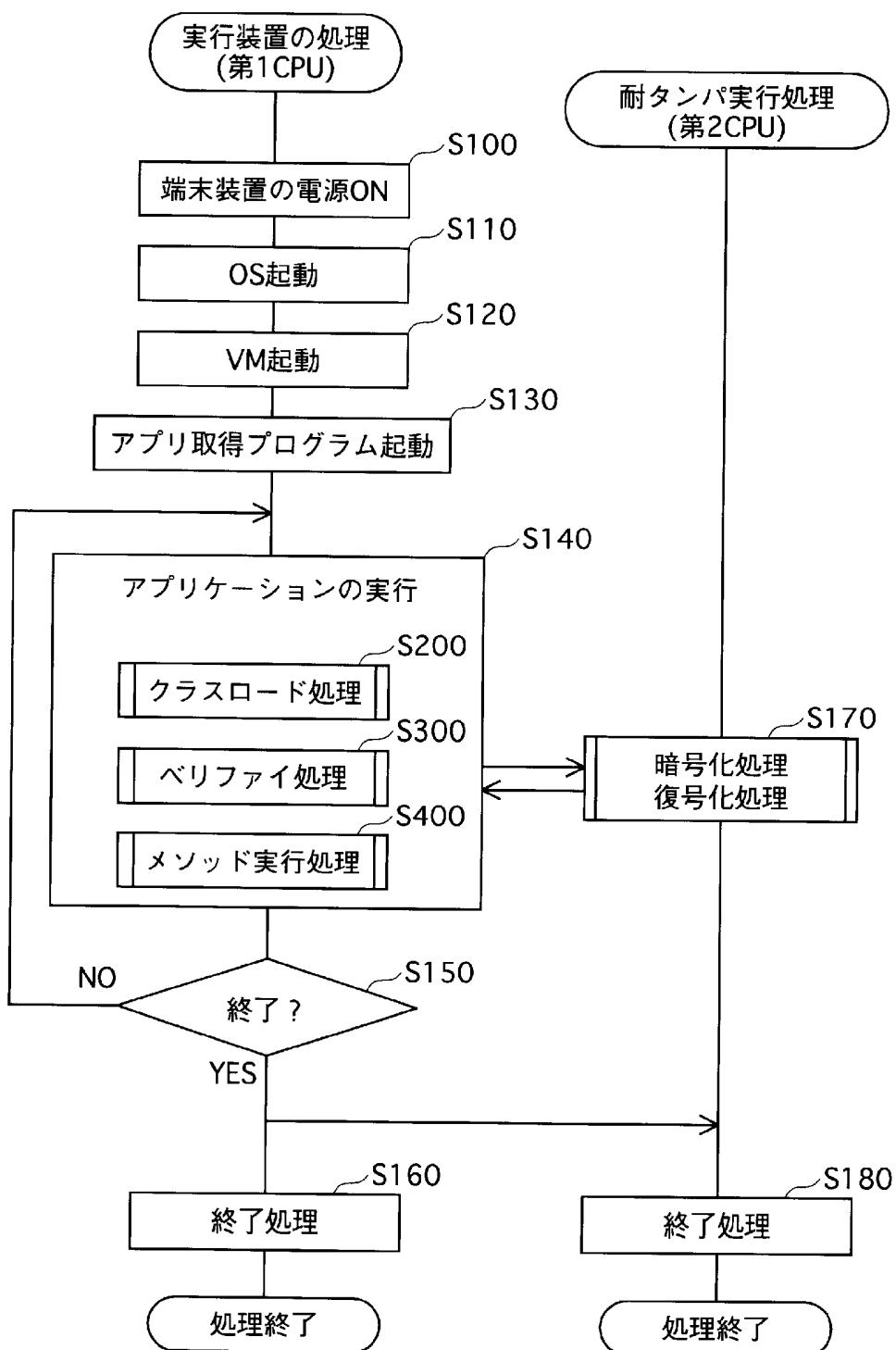
[図6]

鍵ID	暗号鍵	復号鍵	暗号 アルゴリズム
1	fd456	4fgaa	A
2	gf322	442dd	B
3	Ga323	333ff	A
4	135fd	2lji0	C
5	4er34	8lsls	D
6	44423	1llee	B

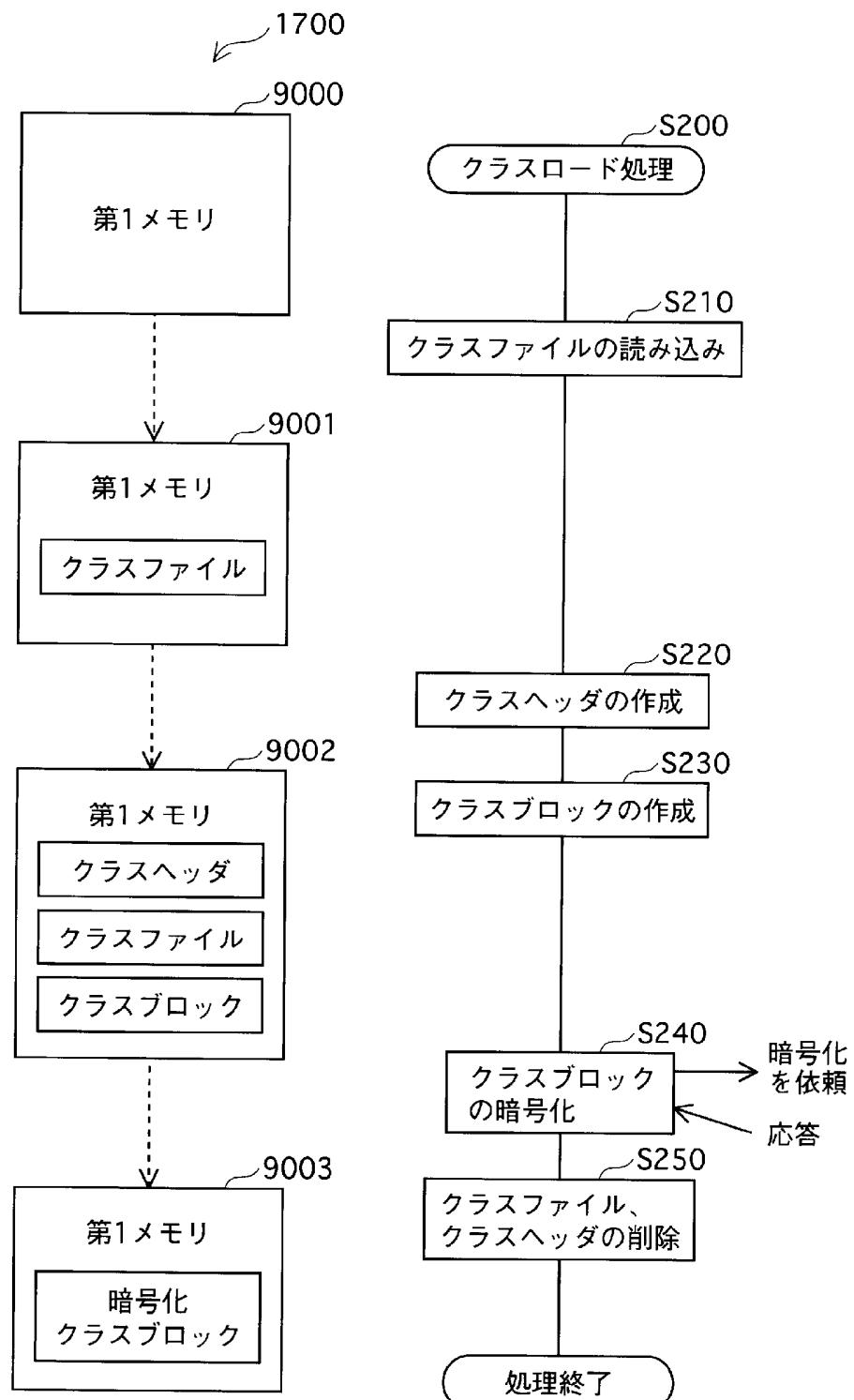
[図7]

アドレス	鍵ID
0x06530012	4
0x0430012d	3
0x0623bc8d	2

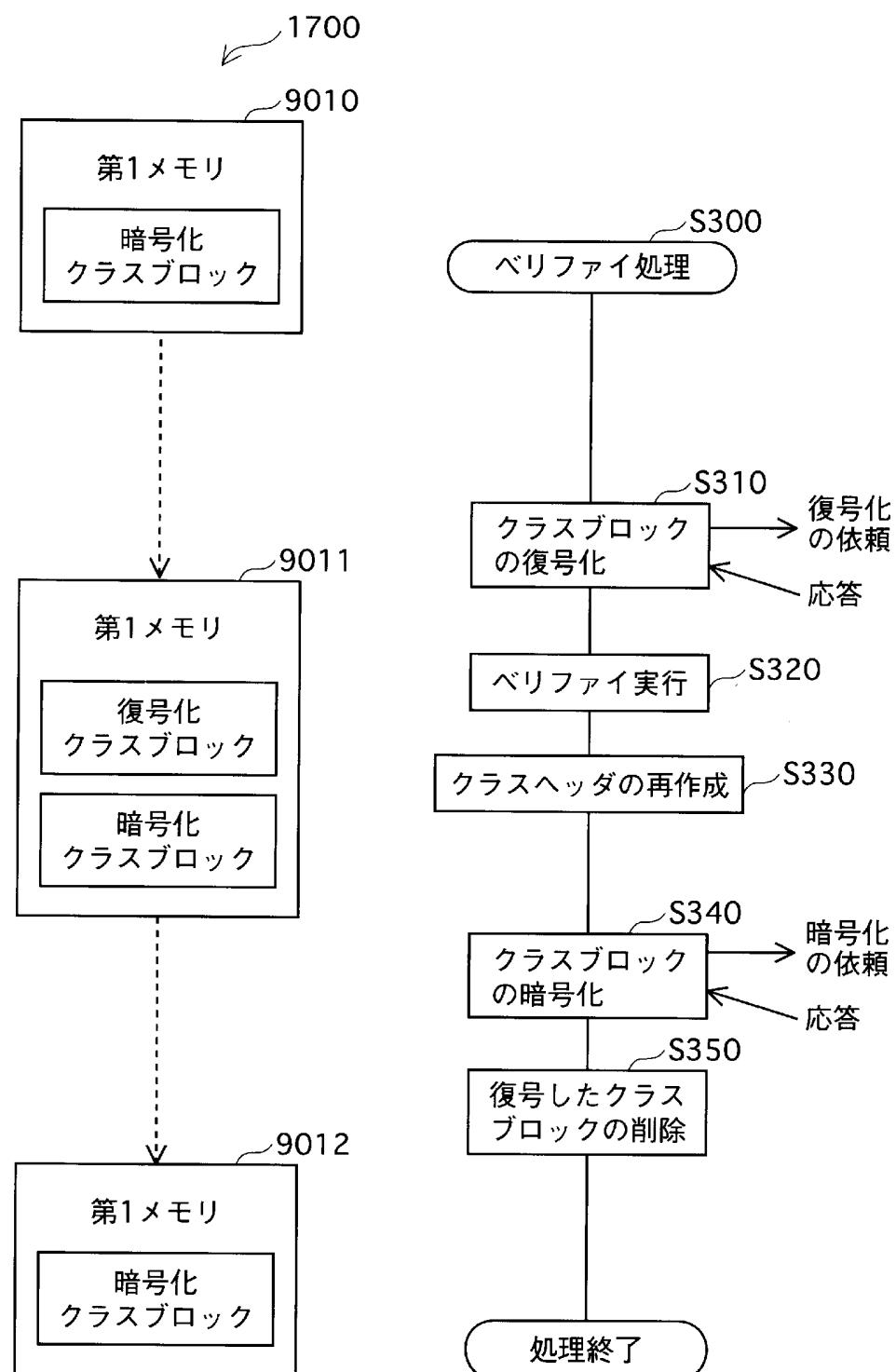
[図8]



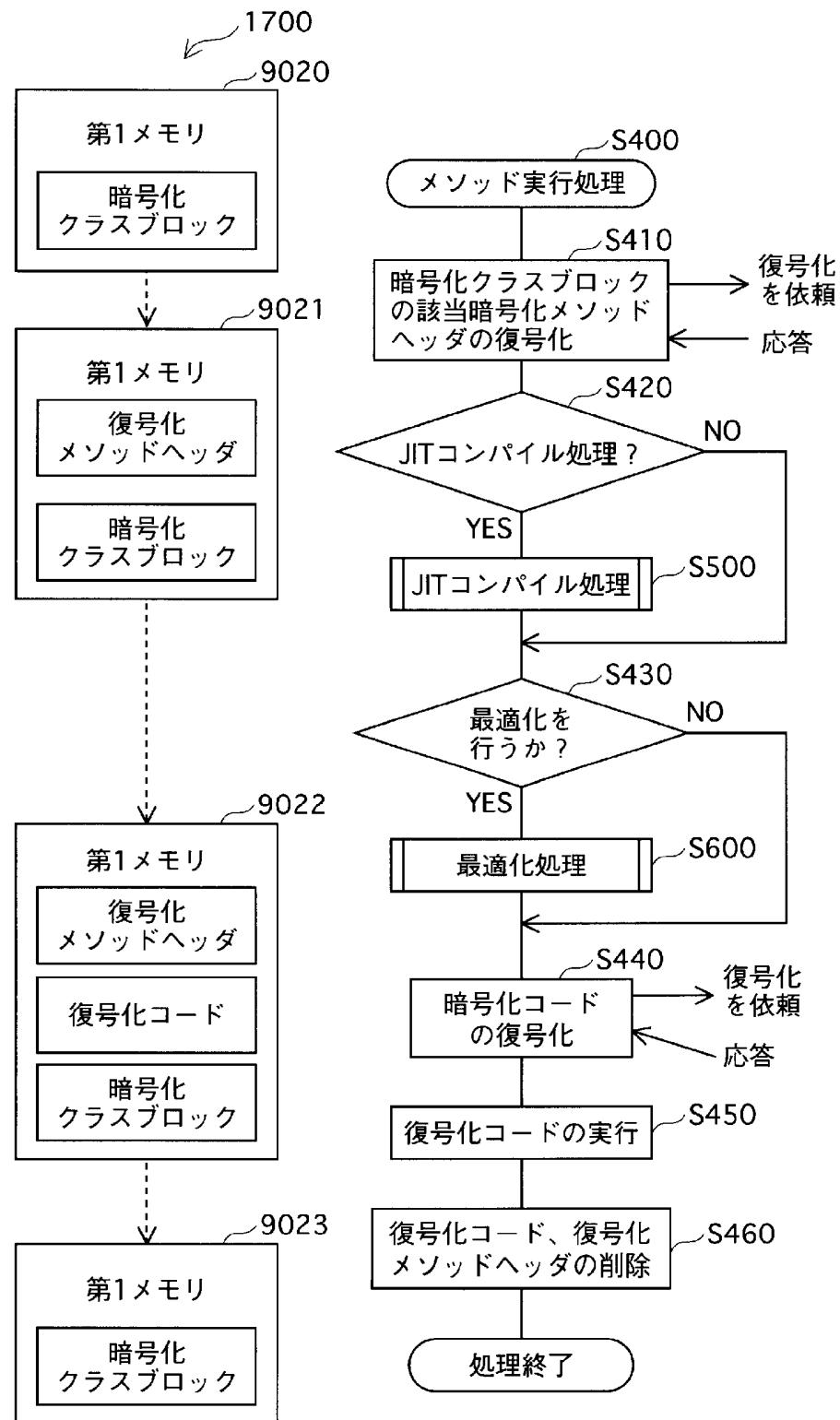
[図9]



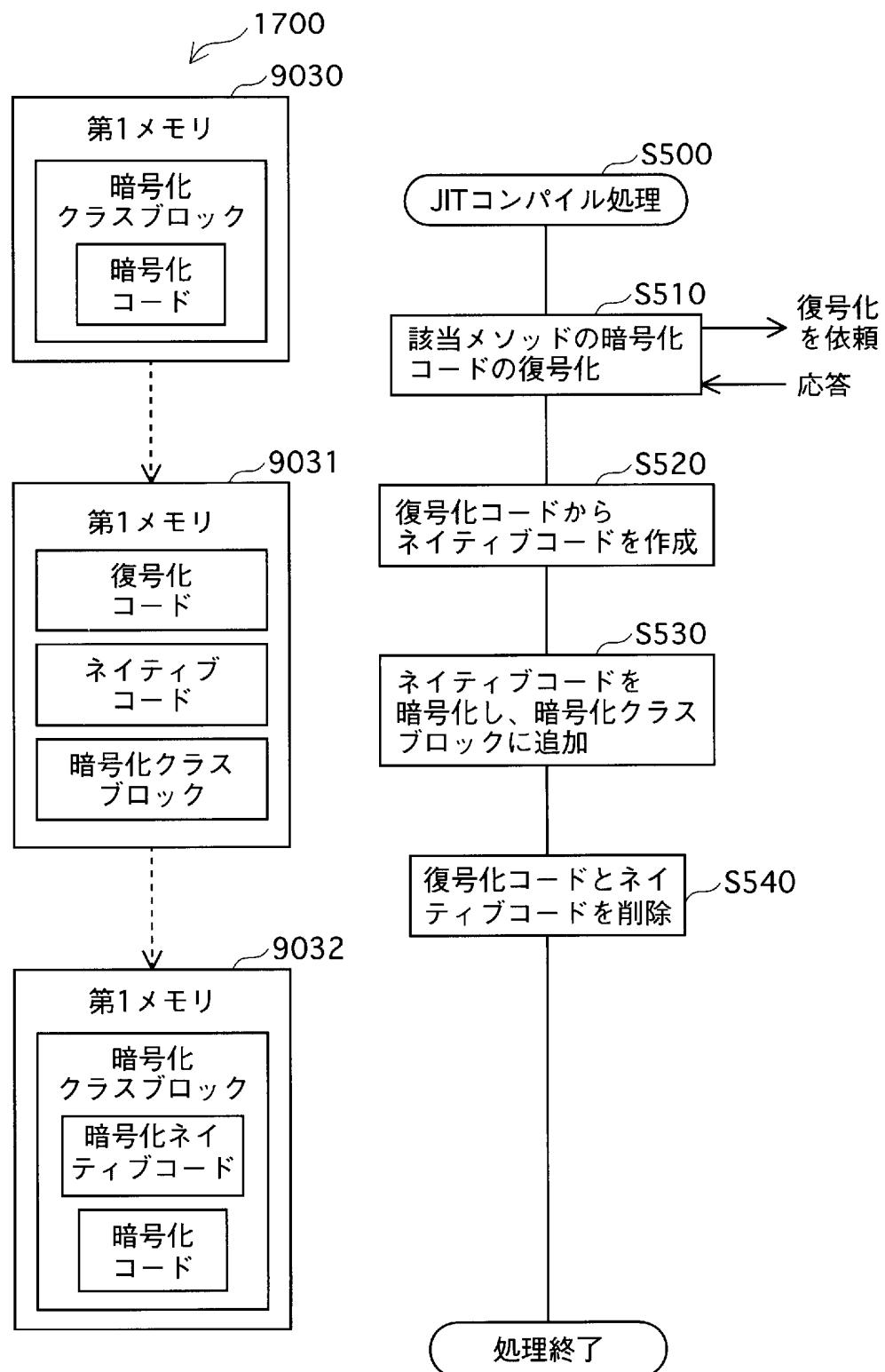
[図10]



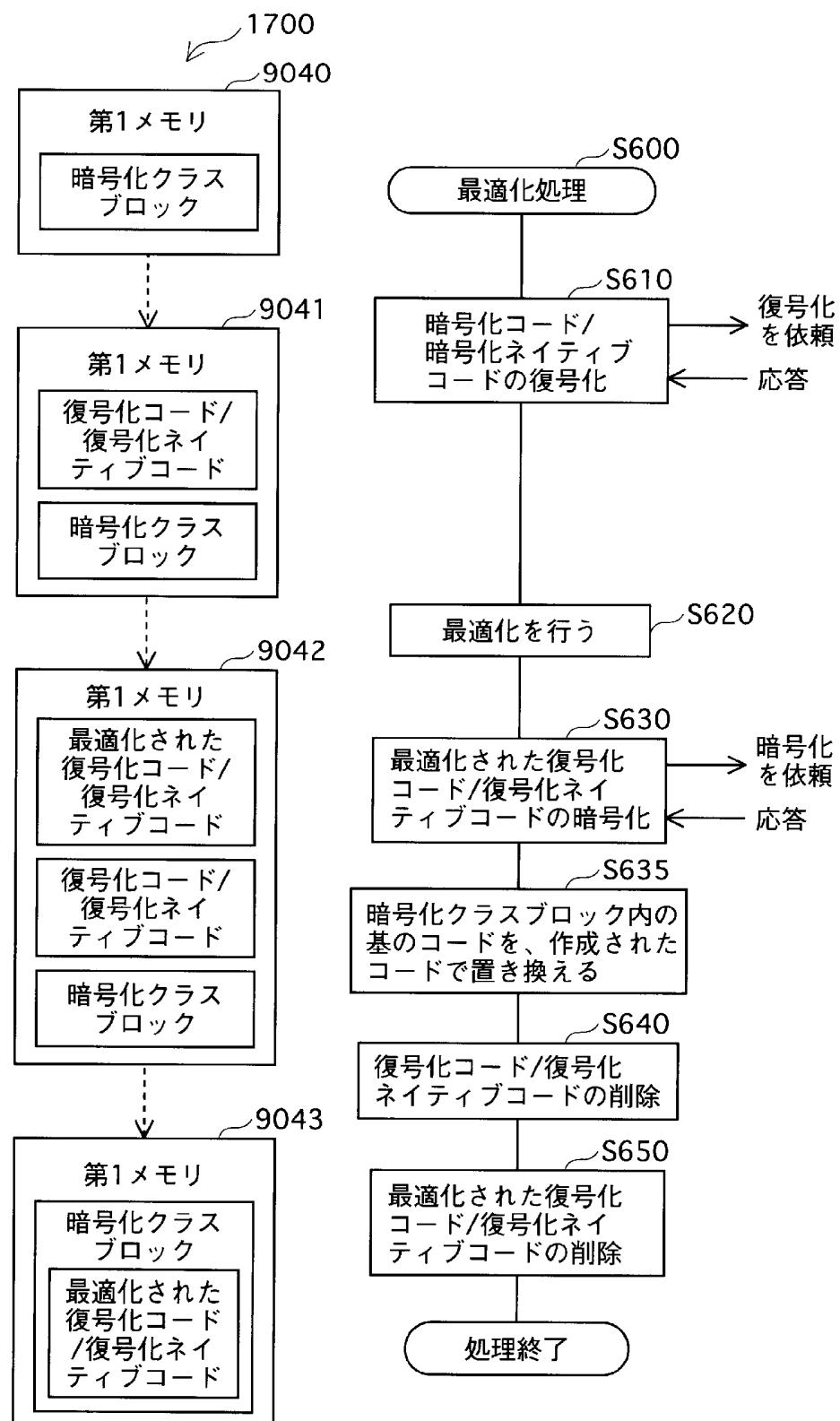
[図11]



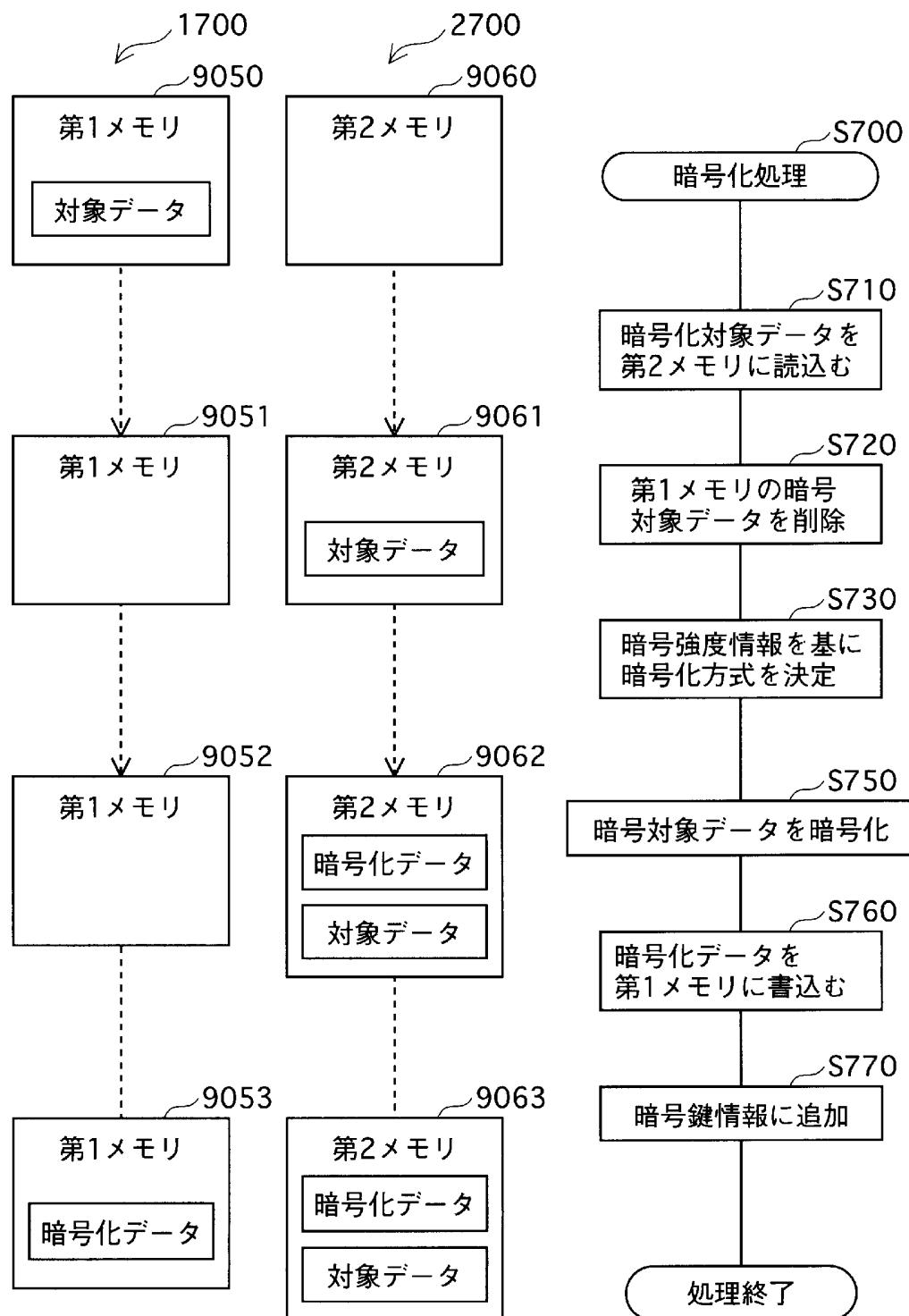
[図12]



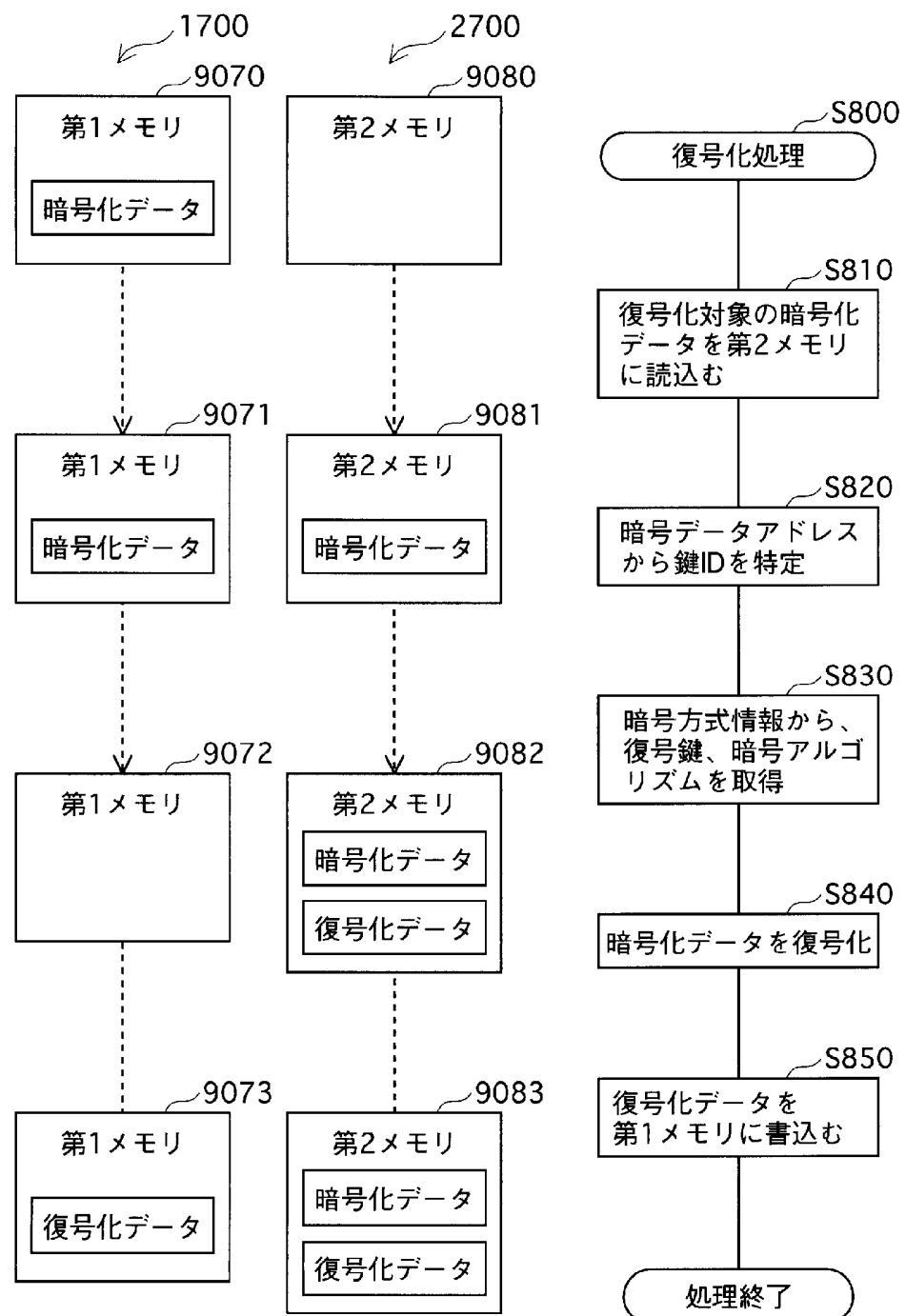
[図13]



[図14]

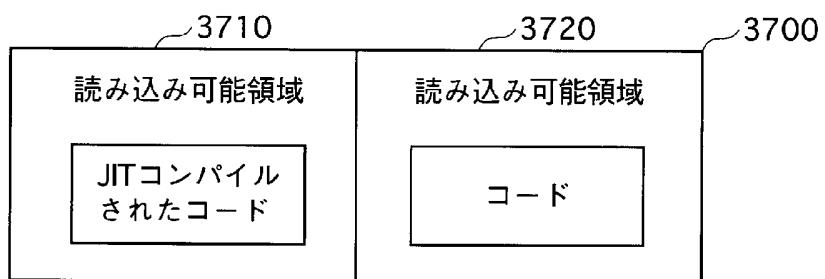


[図15]

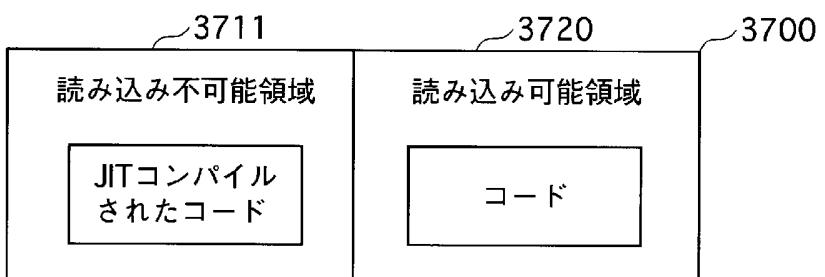


[図16]

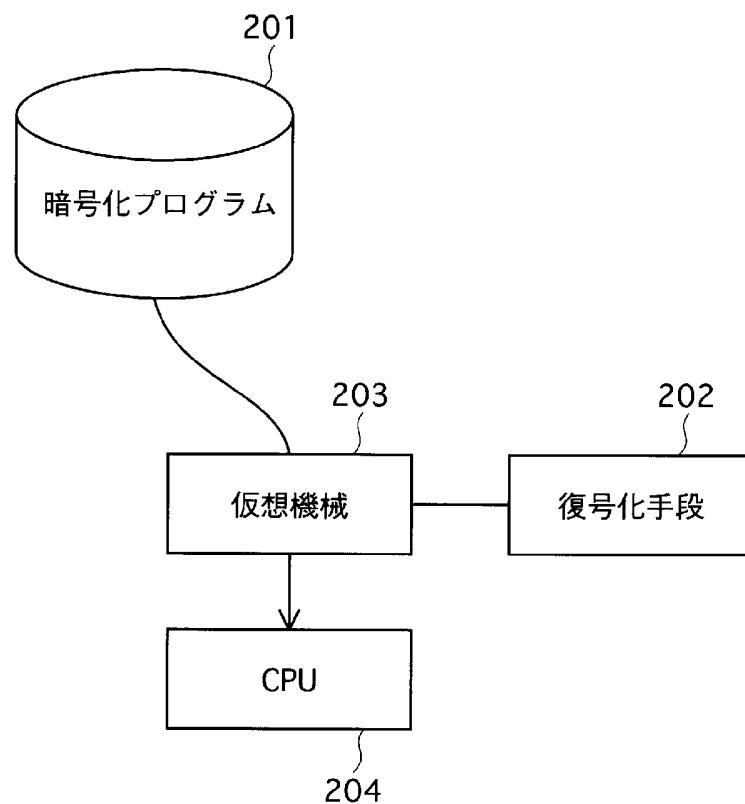
(a)



(b)



[図17]



INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2005/006307

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl⁷ G06F1/00, G06F9/44, G06F12/14

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.Cl⁷ G06F1/00, G06F9/44, G06F12/14

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

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2005
Kokai Jitsuyo Shinan Koho	1971-2005	Toroku Jitsuyo Shinan Koho	1994-2005

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.
X A	JP 2001-175466 A (Fuji Xerox Co., Ltd.), 29 June, 2001 (29.06.01), Full text; all drawings & US 6715142 B1	7 1-6
Y A	Douglas Low. Java Control Flow Obfuscation. [online]. June 1998, pages 10-11. [retrieved on 20 June, 2005 (20.06.05)], Retrieved from the Internet:<URL:http://www.cs.auckland.ac. nz/research/theses/1998/low_douglas_thesis 1998.pdf>	1-6 7
Y A	JP 2002-132364 A (Yutaka IIDUKA), 10 May, 2002 (10.05.02), Full text; all drawings (Family: none)	1-6 7

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
22 June, 2005 (22.06.05)

Date of mailing of the international search report
12 July, 2005 (12.07.05)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2005/006307

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	JP 09-233067 A (Hiroyuki OKANO), 05 September, 1997 (05.09.97), Claim 1 & US 5504818 A	3 1, 2, 4-7

A. 発明の属する分野の分類(国際特許分類(IPC))

Int.Cl.⁷ G06F1/00, G06F9/44, G06F12/14

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.⁷ G06F1/00, G06F9/44, G06F12/14

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	1971-2005年
日本国実用新案登録公報	1996-2005年
日本国登録実用新案公報	1994-2005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 カテゴリーエ	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 請求の範囲の番号
X A	JP 2001-175466 A (富士ゼロックス株式会社) 2001.06.29, 全文, 全図 & US 6715142 B1	7 1-6
Y A	Douglas Low. Java Control Flow Obfuscation. [online]. June 1998, pages 10-11. [retrieved on 2005-06-20]. Retrieved from the Internet:<URL: http://www.cs.auckland.ac.nz/research/theses/1998/low_douglas_thesis1998.pdf>	1-6 7

 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

- 「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 22.06.2005	国際調査報告の発送日 12.7.2005
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) 宮司 卓佳 電話番号 03-3581-1101 内線 3546 5S 9555

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 請求の範囲の番号
Y	JP 2002-132364 A (飯塚豊) 2002.05.10, 全文, 全図	1-6
A	(ファミリーなし)	7
Y	JP 09-233067 A (岡野博一) 1997.09.05, 請求項 1	3
A	& US 5504818 A	1, 2, 4-7