(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7589288号
(P7589288)

(45)発行日　令和6年11月25日(2024.11.25)

(24)登録日　令和6年11月15日(2024.11.15)

(51)国際特許分類
Ｇ０６Ｆ　21/57　(2013.01)

ＦＩ
Ｇ０６Ｆ　21/57　３２０

請求項の数　9　（全16頁）

(21)出願番号　特願2023-81896(P2023-81896)
(22)出願日　令和5年5月17日(2023.5.17)
(62)分割の表示　特願2019-120327(P2019-120327)の分割
原出願日　令和1年6月27日(2019.6.27)
(65)公開番号　特開2023-96096(P2023-96096A)
(43)公開日　令和5年7月6日(2023.7.6)
審査請求日　令和5年6月14日(2023.6.14)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100126240
弁理士　阿部 琢磨
(74)代理人　100223941
弁理士　高橋 佳子
(74)代理人　100159695
弁理士　中辻 七朗
(74)代理人　100172476
弁理士　冨田 一史
(74)代理人　100126974
弁理士　大朋 靖尚
(72)発明者　江口 貴巳
東京都大田区下丸子３丁目３０番２号

最終頁に続く

(54)【発明の名称】　情報処理装置、情報処理方法およびプログラム

(57)【特許請求の範囲】

【請求項１】

任意のタイミングで実行されるソフトウェアを実行に先立って検証する検証手段と、

更新ソフトウェアによる複数のファイルへの書き込みの実行により、ソフトウェアの更新を行う更新手段と、

前記更新手段による複数のファイルへの書き込み開始から終了まで、前記検証手段による検証を制限する制限手段と、を有することを特徴とする情報処理装置。

【請求項２】

前記更新手段は、前記更新ソフトウェアの複数の実行により、ソフトウェアの更新を行うことを特徴とする請求項１に記載の情報処理装置。

【請求項３】

前記更新手段による更新の終了の後、ハードウェアを起点とした方法によって、段階的に起動される複数のソフトウェアを検証する第二の検証手段を更に有することを特徴とする請求項１に記載の情報処理装置。

【請求項４】

前記制限手段は、前記更新手段による更新が行われるソフトウェアの更新のサイズに応じて、前記検証手段による検証を制限することを特徴とする請求項１乃至請求項３のいずれか１項に記載の情報処理装置。

【請求項５】

プリンタ部およびスキャナ部を更に有することを特徴とする請求項１乃至請求項４のい

ずれか１項に記載の情報処理装置。
【請求項６】
前記検証手段の検証によって、前記ソフトウェアの改ざんが検知された場合、当該検知の結果を示す情報を表示させる表示制御手段を更に有することを特徴とする請求項１乃至請求項５のいずれか１項に記載の情報処理装置。
【請求項７】
前記段階的に起動される複数のソフトウェアには、BIOSが含まれることを特徴とする請求項３に記載の情報処理装置。
【請求項８】
検証手段が、任意のタイミングで実行されるソフトウェアを実行に先立って検証する検証工程と、
更新手段が、更新ソフトウェアによる複数のファイルへの書き込みの実行により、ソフトウェアの更新を行う更新工程と、
制限手段が、前記更新手段による複数のファイルへの書き込み開始から終了まで、前記検証手段による検証を制限する制限工程と、を有することを特徴とする情報処理方法。
【請求項９】
コンピュータを、
任意のタイミングで実行されるソフトウェアを実行に先立って検証する検証手段と、
更新ソフトウェアによる複数のファイルへの書き込みの実行により、ソフトウェアの更新を行う更新手段と、
前記更新手段による複数のファイルへの書き込み開始から終了まで、前記検証手段による検証を制限する制限手段と、を有することを特徴とする情報処理装置として機能させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ソフトウェアの検証を行う情報処理装置、情報処理方法およびプログラムに関する。
【背景技術】
【０００２】
情報処理装置を制御するプログラムを第三者が不正な方法で改ざんし、情報処理装置内の情報資産を盗む攻撃や、プログラムが改ざんされた情報処理装置を踏み台に利用する攻撃が問題になっている。このような攻撃を防止するために、情報処理装置内にプログラムが第三者によって改ざんされていないことを検証する方法が考案されている。
【０００３】
ユーザが情報処理装置によって実現される機能を利用するときに情報処理装置内のプログラムが改ざんされていないことを保証するためには、その機能を実現するプログラムの実行の直前に検証する必要がある。これは、プログラムの検証と実行の間に時間差があると、その間に改ざんされる場合に対応できないためである。
【０００４】
プログラムの実行直前にプログラムを検証する方法の例として、特許文献１では、プログラムの実行要求をフックするフッキングプログラム（実行時検証プログラム）とホワイトリストを用いる方法が提案されている。ホワイトリストとは、起動が許可されたプログラムの一覧を示すリストである。この方法では、プログラム起動前に、プログラムのハッシュ値を計算してホワイトリストと比較し、一致する場合のみプログラムを起動する。プログラムの検証に失敗した場合、プログラムの実行要求に対してエラーを返す。この時、プログラムの実行要求に対する一般的なエラーによりプロセスにエラーを返す。ホワイトリストがない場合、プログラムの検証時に全プログラムがホワイトリストに記載されていないとみなされるため、全プログラムの検証に失敗し実行させないので、プログラムの検証を有効化する前にホワイトリストの作成が必須である。

【０００５】
　また、起動を許可したプログラムに対し、不揮発性メモリや外部記憶装置への書き込みを許可する書き込み制御を行うことも可能である。
【０００６】
　この方法では、マルチプロセスでプログラムを実行する情報処理装置において、プログラムを実行する各プロセスがプログラムを実行する度にプログラムの検証を行う必要がなく、フッキングプログラムに検証を任せることができる。
【先行技術文献】
【特許文献】
【０００７】
【文献】特願２０１８－０８０７７５号公報
【発明の開示】
【発明が解決しようとする課題】
【０００８】
　従来技術では、実行時検証機能が有効になっている状態でソフトウェア更新を行うと、ソフトウェア更新プログラムに対しても 実行時検証機能が書き込み制御を行う。そのため、ソフトウェア更新プログラムが検証対象となり、ソフトウェア更新に通常の数倍の時間がかかる。
【０００９】
　そこで、本発明は、実行時検証機能が有効になっている状態でソフトウェア更新する際には書き込み制御を無効にし、かつソフトウェア更新終了後に再起動し起動時検証を行うことにより、セキュリティを維持しつつ速度の低下を防ぐことを目的とする。
【課題を解決するための手段】
【００１０】
　本発明は、任意のタイミングで実行されるソフトウェアを実行に先立って検証する検証手段と、更新ソフトウェアによる複数のファイルへの書き込みの実行により、ソフトウェアの更新を行う更新手段と、前記更新手段による複数のファイルへの書き込み開始から終了まで、前記検証手段による検証を制限する制限手段と、を有することを特徴とする。
【発明の効果】
【００１１】
　本発明によれば、本発明では、実行時検証機能が有効になっている状態でソフトウェア更新時に書き込み制御を無効にし、かつソフトウェア更新終了後に再起動し起動時検証を行うことにより、セキュリティを維持しつつ速度の低下を防ぐことができる。
【図面の簡単な説明】
【００１２】
【図１】ＭＦＰとクライアントＰＣの接続形態を示すブロック図である。
【図２】ＭＦＰのコントローラ部の内部構成図である。
【図３】ＭＦＰのコントローラ内で実行されるソフトウェアのブロック図である。
【図４】正解値リスト、ホワイトリストの例である。
【図５】設定に係る画面構成図である。
【図６】設定に係る画面構成図である。
【図７】設定に係る画面構成図である。
【図８】ＭＦＰ側の処理を実施するフロー図である。
【図９】ＭＦＰ側の処理を実施するフロー図である。
【図１０】パラメータ間の関係を示すグラフである。
【図１１】本発明のＭＦＰ側の処理を実施するフロー図である。
【発明を実施するための形態】
【００１３】
（第１の実施形態）
　以下、本実施形態を図面に基づいて解説する。本実施形態では、ソフトウェア更新処理

、および機能実行時のソフトウェア検証処理について説明する。ここではＭＦＰ（Ｍｕｌｔｉ－Ｆｕｎｃｔｉｏｎ　Ｐｅｒｉｐｈｅｒａｌ：複合機）を例に実施形態を説明するが、本発明は複合機以外の任意の情報処理装置に適用可能な技術である。

【００１４】

図１はＭＦＰとクライアントＰＣの接続形態を示すブロック図である。ＭＦＰ１００とクライアントＰＣ１２０はＬＡＮ１５０を介して接続されている。ＭＦＰ１００はユーザとの入出力を行う操作部１０２を有する。ＭＦＰ１００は電子データを紙媒体に出力するプリンタ部１０３を有する。ＭＦＰ１００は紙媒体を読み込み電子データに変換するスキャナ部１０４を有する。操作部１０２とプリンタ部１０３とスキャナ部１０４はコントローラ部１０１に接続され、コントローラ部１０１の制御に従い複合機としての機能を実現する。クライアントＰＣ１２０はＭＦＰ１００に対してプリントジョブの送信といった処理を行う。

【００１５】

図２はＭＦＰのコントローラ部１０１の詳細を示すブロック図である。ＣＰＵ２０１はコントローラ内の主な演算処理を行う。ＣＰＵ２０１はバスを介してＤＲＡＭ２０２と接続される。ＤＲＡＭ２０２はＣＰＵ２０１が演算する過程で演算命令を表すプログラムデータや、処理対象のデータを一時的に配置するための作業メモリとしてＣＰＵ２０１によって使用される。ＣＰＵ２０１はバスを介してＩ／Ｏコントローラ２０３と接続される。Ｉ／Ｏコントローラ２０３はＣＰＵ２０１の指示に従い各種デバイスに対する入出力を行う。Ｉ／Ｏコントローラ２０３にはＳＡＴＡ（Ｓｅｒｉａｌ　Ａｄｖａｎｃｅｄ　Ｔｅｃｈｎｏｌｏｇｙ　Ａｔｔａｃｈｍｅｎｔ）Ｉ／Ｆ２０５が接続され、その先にＦｌａｓｈＲＯＭ２１１が接続される。ＣＰＵ２０１はＦｌａｓｈＲＯＭ２１１をＭＦＰの機能を実現するためのプログラム、およびドキュメントファイルを永続的に記憶するために使用する。Ｉ／Ｏコントローラ２０３にはネットワークＩ／Ｆ２０４が接続され。ネットワークＩ／Ｆ２０４の先には、有線ＬＡＮデバイス２１０が接続される。ＣＰＵ２０１はネットワークＩ／Ｆ２０４を介して有線ＬＡＮデバイス２１０を制御することで、ＬＡＮ１５０上の通信を実現する。Ｉ／Ｏコントローラ２０３にはパネルＩ／Ｆ２０６が接続され、ＣＰＵ２０１はパネルＩ／Ｆ２０６を介して操作部１０２に対するユーザ向けの入出力を実現する。Ｉ／Ｏコントローラ２０３にはプリンタＩ／Ｆ２０７が接続され、ＣＰＵ２０１はプリンタＩ／Ｆ２０７を介してプリンタ部１０３を利用した紙媒体の出力処理を実現する。Ｉ／Ｏコントローラ２０３にはスキャナＩ／Ｆ２０８が接続され、ＣＰＵ２０１はスキャナＩ／Ｆ２０８を介してスキャナ部１０４を利用した原稿の読み込み処理を実現する。Ｉ／Ｏコントローラ２０３にはＵＳＢ　Ｉ／Ｆ２０９が接続され、ＵＳＢ　Ｉ／Ｆに接続された任意の機器の制御を行う。ＲＯＭ２２０はＣＰＵ２０１とバスで接続されていて、ＢＩＯＳ（Ｂａｓｉｃ　Ｉｎｐｕｔ　Ｏｕｔｐｕｔ　Ｓｙｓｔｅｍ）を実現する制御プログラムを記憶している。ＢＩＯＳ検証ユニット２２１はＲＯＭ２２０およびＣＰＵ２０１とバスで接続されていて、ＲＯＭ２２０に記憶されたＢＩＯＳデータの検証と、ＣＰＵへのＢＩＯＳ起動指示を行う。ここで、ＢＩＯＳ検証ユニット２２１はハードウェアであることを明記し、ＢＩＯＳ検証がハードウェア検証であることを確認しておく。ＢＩＯＳ検証ユニット２２１とＣＰＵ２０１を繋ぐバスは悪意のある第三者に細工をされないために、同一チップ、またはそれに準ずる構成で実現され外部から物理的に確認できない形態になっている。本実施形態では、ＢＩＯＳ検証ユニット２２１の制御機構は集積回路としてハードウェアで実現されている構成を想定するが、専用のＣＰＵ、制御ソフトを記憶したＲＯＭといった要素を同一チップ内に実装し、製造後に変更できない構成であっても良い。

【００１６】

コピー機能を実施する場合は、ＣＰＵ２０１がＳＡＴＡ　Ｉ／Ｆ２０５を介してＦｌａｓｈＲＯＭ２１１からプログラムデータをＤＲＡＭ２０２に読み込む。ＣＰＵ２０１がＤＲＡＭ２０２に読み込まれたプログラムに従いパネルＩ／Ｆ２０６を介して操作部１０２に対するユーザからのコピー指示を検出する。ＣＰＵ２０１はコピー指示を検出するとスキャナＩ／Ｆ２０８を介してスキャナ部１０４から原稿を電子データとして受け取りＤＲ

ＡＭ２０２に格納する。ＣＰＵ２０１はＤＲＡＭ２０２に格納した画像データに対して出力に適した色変換処理などを実施する。ＣＰＵ２０１はＤＲＡＭ２０２に格納した画像データをプリンタＩ／Ｆ２０７を介してプリンタ部１０３に転送し、紙媒体への出力処理を実施する。

【００１７】

ＰＤＬ印刷を実施する場合は、クライアントＰＣ１２０がＬＡＮ１５０を介して印刷指示を行う。ＣＰＵ２０１はＳＡＴＡ　Ｉ／Ｆ２０５を介してＦｌａｓｈＲＯＭ２１１からプログラムデータをＤＲＡＭ２０２に読み込み、ＤＲＡＭ２０２に読み込まれたプログラムに従いネットワークＩ／Ｆ２０４を介して印刷指示を検出する。ＣＰＵ２０１はＰＤＬ送信指示を検出するとネットワークＩ／Ｆ２０４を介して印刷データを受信し、ＳＡＴＡＩ／Ｆ２０５を介してＦｌａｓｈＲＯＭ２１１に印刷データを保存する。ＣＰＵ２０１は印刷データの保存が完了すると、ＦｌａｓｈＲＯＭ２１１に保存した印刷データをＤＲＡＭ２０２に画像データとして展開する。ＣＰＵ２０１はＤＲＡＭ２０２に格納した画像データに対して出力に適した色変換処理などを実施する。ＣＰＵ２０１はＤＲＡＭ２０２に格納した画像データをプリンタＩ／Ｆ２０７を介してプリンタ部１０３に転送し、紙媒体への出力処理を実施する。

【００１８】

図３はＭＦＰのコントローラ部１０１で実行されるソフトウェアの構造をあらわすブロック図である。コントローラ部１０１で実行されるソフトウェアは全て、ＣＰＵ２０１が実行する。ＣＰＵ２０１は、ＲＯＭ２２０に記憶されたＢＩＯＳ３６０を実行する。ＣＰＵ２０１は、ＦｌａｓｈＲＯＭ２１１に記憶された、ローダ３７０、Ｉｎｉｔｒｄ３８０、コントローラソフト３００をＤＲＡＭ２０２に読み込んだ後に実行する。ＢＩＯＳ３６０はＩ／Ｏコントローラ２０３やＤＲＡＭ２０２をＣＰＵ２０１が制御するための基本処理を実行する。ＢＩＯＳ３６０は内部的にＢＩＯＳとしての制御ソフトと制御ソフトに対応する署名データで構成されている。ローダ読み込み検証部３６１はＢＩＯＳ３６０の制御ソフトに含まれ、ローダを検証する処理とローダに付与された署名に対応する公開鍵を含む。さらにＢＩＯＳ３６０はＦｌａｓｈＲＯＭ２１１からローダ３７０を読み込み、開始する処理を含む。ローダ３７０はＦｌａｓｈＲＯＭ２１１からカーネル３９０、Ｉｎｉｔｒｄ３８０を読み込み、開始する処理を実行する。ローダ３７０は内部的にローダとしての制御ソフトと制御ソフトに対応する署名データで構成されている。カーネル、Ｉｎｉｔｒｄ読み込み検証部３７１はローダ３７０に含まれ、カーネル、Ｉｎｉｔｒｄを検証する処理とカーネル、Ｉｎｉｔｒｄに付与された署名に対する公開鍵を含む。Ｉｎｉｔｒｄ３８０はＦｌａｓｈＲＯＭ２１１からコントローラソフト３００を読み込み、開始する処理を実行する。Ｉｎｉｔｒｄ３８０は内部的にＩｎｉｔｒｄとしての制御ソフトと制御ソフトに対する署名データで構成されている。起動時検証部３８１はＩｎｉｔｒｄ３８０に含まれ、コントローラソフト３００を構成する、全てのプログラムファイルを起動時に検証する処理と、付与された署名に対する公開鍵を含む。ここで、全ての署名データに対する秘密鍵はソフトウェアの開発時のみ利用され一般に流通することはない。

【００１９】

操作制御部３０１は操作部１０２にユーザ向けの画面イメージを表示、およびユーザ操作の検知と画面上に表示したボタン等の画面部品に紐づけられた処理を実行する。データ記憶部３０２は他の制御部からの要求でデータをＦｌａｓｈＲＯＭ２１１に記憶、および読み出しを行う。例えば、ユーザが何らかの機器設定を変更したい場合は、操作部１０２にユーザが入力した内容を操作制御部３０１が検知し、操作制御部３０１からの要求でデータ記憶部３０２が設定値としてＦｌａｓｈＲＯＭ２１１に保存する。ネットワーク制御部３０７はデータ記憶部３０２に記憶された設定値に従い、システム起動時や、設定変更検出時にＩＰアドレスなどネットワーク設定をＴＣＰ／ＩＰ制御部３０８に行う。ＴＣＰ／ＩＰ制御部３０８は他の制御からの指示に従い、ネットワークＩ／Ｆ２０４を介して、ネットワークパケットの送受信処理を行う。ジョブ制御部３０３は他の制御部からの指示に従って、ジョブ実行の制御を行う。画像処理部３０４はジョブ制御部３０３からの指示

に従って、画像データを用途ごとに適した形式に加工する。印刷処理部３０５はジョブ制御部３０３からの指示に従い、プリンタＩ／Ｆ２０７を介して、紙媒体に画像を印刷し出力する。読み取り制御部３０６はジョブ制御部３０３からの指示に従い、スキャナＩ／Ｆ２０８を介して、設置された原稿を読み込む。認証部３０９は管理者権限が必要な操作に対して、操作者が管理者であるか否かを判断する処理を行う。ソフトウェア更新部３１０はコントローラソフト３００を構成するプログラムファイルを、設置環境で更新する処理を行う。ＵＳＢ制御部３１１はＵＳＢ　Ｉ／Ｆ２０９を制御し、ＵＳＢ接続された任意の機器の制御を行う。起動時検証用正解値リスト３２１は起動検証部３８１が検証処理に利用する正解値のリストである。実行時検証部３２２は、コントローラソフト３００を構成する、全てのプログラムファイルを実行時に検証する処理を含む。実行時検証用ホワイトリスト３２３は実行時検証部３２２が検証処理に利用する正解値のリストである。

【００２０】

例えば、コピー機能を実行する場合は操作制御部３０１がコピー機能の開始要求を検知し、ジョブ制御部３０３にコピーを指示する。ジョブ制御部３０３は読み取り制御部３０６に原稿読み取りを指示し、スキャン画像を取得する。ジョブ制御部３０３は画像処理部３０５に指示し、スキャン画像を印刷に適した形式に変換する。ジョブ制御部３０３は印刷制御部３０５に印刷を指示し、コピー結果を出力する。

【００２１】

図４は起動時検証用正解値リスト３２１、および実行時検証用ホワイトリスト３２３のデータ形式のサンプルである。コントローラソフト３００に含まれる全てのプログラムファイルに対して、ファイル名３００１とハッシュ３００２の組み合わせをリスト化したものである。データの内容としては、少なくともファイル名称、ファイルの配置場所（ディレクトリ上の位置）、ファイルから計算したハッシュ値を含むものとする。起動時検証用正解値リスト３２１と実行時検証用ホワイトリスト３２３は同じファイルとすることも可能であるが、生成タイミングの違い、利用形態の違いから、ここでは別のファイルとして配置する。

【００２２】

図５は操作部１０２に表示される、メニュー画面４０１であり複合機が持つさまざまな機能の実行をユーザが指示するためのものである。ボタン４０２はコピー機能をユーザが指示するために利用される。ボタン４０３はスキャンして保存する機能をユーザが指示するために利用される。ボタン４０４はスキャンして送信する機能をユーザが指示するために利用される。ボタン４０５は機器の設定変更をユーザが指示するために利用される。ボタン４０５を押すことで、設定画面５０１を表示することができる。表示領域４０６は機器の動作中に発生したさまざまなユーザ向けのメッセージを表示する。

【００２３】

図６は操作部１０２に表示される、設定画面５０１でありさまざまな設定をユーザが指示するためのものである。この画面自体に具体的な設定項目はなく、詳細な設定項目へのガイドとなる中間階層である。ボタン５０２を押すことで不図示のセキュリティ設定画面を表示することができる。ボタン５０３を押すことで、不図示の機器設定画面を表示することができる。ボタン５０４を押すことで、不図示のユーザ設定画面を表示することができる。ボタン５０５を押すことで、ソフトウェアの更新を開始することができる。表示領域５０６は機器の動作中に発生したさまざまなユーザ向けのメッセージを表示する。

【００２４】

図７は操作部１０２に表示される、エラー画面８０１である。この画面はシステム停止エラーとしてユーザにファームウェアが改ざんされ、システムを停止したことを通知している。また、この画面から通常の機能実行画面に遷移することはできず、ユーザはＭＦＰ１００を利用することはない。

【００２５】

図８を用いて、ＭＦＰ１００がソフトウェアを更新する際の実行時検証機能と起動時検証機能の処理フローを説明する。

【００２６】
図８のＭＦＰ１００が実施する更新処理は、ＦｌａｓｈＲＯＭ２１１に格納されたプログラムをＣＰＵ２０１がＤＲＡＭ２０２に読み込んだのち、ＣＰＵ２０１の演算処理として実施するものである。
【００２７】
ボタン５０５が押下され、ソフトウェア更新が指示された後、ＭＦＰ１００はＳ８０１としてソフトウェアの更新処理を開始する。ソフトウェア更新処理はソフトウェア更新部３１０が行う。
【００２８】
Ｓ８０２でＭＦＰ１００はＵＳＢ制御部３１１を利用してＵＳＢストレージが接続済みであるか確認する。確認が取れた場合はＳ８０３を実行し、取れなかった場合はＳ８０４を実行する。Ｓ８０４でＭＦＰ１００は操作部１０２に表示される、メッセージ表示領域５０６にエラーメッセージを表示する。
【００２９】
Ｓ８０３でＭＦＰ１００はＵＳＢ制御部３１１を利用してＵＳＢストレージに更新用のファイルが存在するか確認する。確認できた場合はＳ８０５を実行し、確認できなかった場合はＳ８０４を実行する。この時、ファイル名やファイルフォーマット、ファイルの署名を検証するなどして、正当な更新用ファイルであることを確認しても良い。
【００３０】
Ｓ８０５でＭＦＰ１００は実行時検証機能が動作中か確認する。もし動作中でなければ、Ｓ８０７を実行する。動作中であれば、Ｓ８０６を実行する。Ｓ８０６でＭＦＰ１００は実行時検証機能を停止する。実行時検証機能を停止する理由は、通常、更新するプログラムファイルは数百に及ぶことから、書き込みが起きるたびに実行時検証用ホワイトリスト３２３を用いたソフトウェア更新部３１０の検証が行われ、ソフトウェアの更新に時間がかかるためである。
【００３１】
なお、実行時検証機能のうち、書き込み制御機能のみ停止してもよい。
【００３２】
Ｓ８０７でＭＦＰ１００はＵＳＢ制御部３１１を利用してＵＳＢストレージに在る更新用ファイルを、データ記憶部３０２を利用してＦｌａｓｈＲＯＭ２１１内の一時領域に展開する。
【００３３】
Ｓ８０８でＭＦＰ１００はデータ記憶部３０２を利用して、更新用ファイルに含まれる新しい起動時検証用正解値リスト３２１をＦｌａｓｈＲＯＭ２１１の起動時検証用正解リスト格納領域に反映する。
【００３４】
Ｓ８０９でＭＦＰ１００はデータ記憶部３０２を利用して、更新用ファイルに含まれるプログラムファイルをＦｌａｓｈＲＯＭ２１１のプログラム格納領域に反映する。この時反映されるプログラムファイルは、変更が必要な差分のファイルであっても良いし、変更のないファイルを含めた全ファイルであっても良い。
【００３５】
Ｓ８１０でＭＦＰ１００は実行時検証部３２２を利用して、Ｓ８０９で反映したプログラムファイルを実行時検証用ホワイトリスト３２３に反映する。実行時検証用ホワイトリスト３２３はプログラムファイルのファイル名、ファイルパス、ハッシュ値を含むデータである。また、拡張機能をアプリケーションとして追加可能な機器である場合は、ソフトウェア更新部３１０が、アプリケーションを追加したタイミングで実行時検証用ホワイトリスト３２３にアプリケーションに対応する情報を追加する。こうすることで、アプリケーションを後から追加可能な機器であっても、ソフトウェアの改ざんを検知することができる。
【００３６】

Ｓ８１１でＭＦＰ１００は実行時検証機能を開始する。
【００３７】
Ｓ８１２でＭＦＰ１００はシステムを再起動する。
【００３８】
Ｓ８１３以降は、ＭＦＰ１００が起動時にソフトウェアを検証する処理フローである。この処理は、起動するたびに一度行われる。Ｓ８１３でＭＦＰ１００が実施する処理は、ＢＩＯＳ検証ユニット２２１が実施するものである。以下の説明で、Ｓ８１３の検証処理をハードウェア検証と呼ぶ。Ｓ８１７以降でＭＦＰ１００が実施する処理は、ＦｌａｓｈＲＯＭ２１１に格納されたプログラムをＣＰＵ２０１がＤＲＡＭ２０２に読み込んだのち、ＣＰＵ２０１の演算処理として実施するものである。以下の説明で、Ｓ８１７以降の部分ソフトウェアの段階的な検証処理をソフトウェア検証と呼ぶ。夫々の検証処理は同じＭＦＰ１００による検証処理であっても、検証主体が異なること、ハードウェア検証はＣＰＵ２０１の実行するソフトウェアの検証処理ではないことに留意されたい。
【００３９】
再起動処理が開始されるとＢＩＯＳ検証ユニット２２１が起動され、Ｓ８１３としてＢＩＯＳの検証処理を開始する。
【００４０】
Ｓ８１３でＭＦＰ１００はＢＩＯＳ３６０の検証処理を実施し、成功したかどうか確認する。成功した場合はＳ８１６を実行し、失敗した場合はＳ８１４を実行する。検証処理としてはＢＩＯＳ検証ユニット２１１がＲＯＭ２２０から読み込んだＢＩＯＳ３６０の署名に対して、ＢＩＯＳ検証ユニット２１１に配置された公開鍵を用いて署名検証を行う。本実施形態における起動時検証は、起動順序を考慮した署名検証であり、署名検証主体は次に起動する主体の署名検証を行うことでセキュリティ性を担保する。
【００４１】
Ｓ８１６でＭＦＰ１００はＣＰＵ２０１に指示することでＢＩＯＳ３６０を起動する。
【００４２】
Ｓ８１４でＭＦＰ１００内において動作するＢＩＯＳ検証ユニット２２１はユーザ通知に関するデバイスを持たないため、ＬＥＤ（Ｌｉｇｈｔ Ｅｍｉｔｔｉｎｇ Ｄｉｏｄｅ）を発光させることで通知を行い、Ｓ８１５を実行する。
【００４３】
Ｓ８１５でＭＦＰ１００はＢＩＯＳの起動を行わず、起動シーケンスをこのステップで中止することでシステムを停止する。
【００４４】
ハードウェア検証は、ハードウェアで実装された検証方法であり、この検証処理を改ざんするためには集積回路の改ざんが必要であり、極めて堅牢な検証方法である。
【００４５】
ＢＩＯＳ３６０が起動されると、Ｓ８１７としてＦｌａｓｈＲＯＭ２１１に配置されたソフトウェアの検証処理を開始する。
【００４６】
Ｓ８１７でＭＦＰ１００はローダ読み込み検証部３６１を利用して、ローダ３７０の検証処理を実施し、成功したかどうか確認する。成功した場合はＳ８１８を実行し、失敗した場合はＳ８１４を実行する。検証処理としてはＦｌａｓｈＲＯＭ２１１から読み込んだ、次の起動対象であるローダ３７０の署名に対して、ローダ読み込み検証部３６１が持つ公開鍵を用いて署名検証を行う。
【００４７】
Ｓ８１８でＭＦＰ１００はローダを起動する。
【００４８】
Ｓ８１９でＭＦＰ１００はカーネル、Ｉｎｉｔｒｄ読み込み検証部３７１を利用して、カーネル３９０の検証処理を実施し、成功したかどうか確認する。成功した場合はＳ８２０を実行し、失敗した場合はＳ８１４を実行する。検証処理としてはＦｌａｓｈＲＯＭ２

１１から読み込んだ、次の起動対象であるカーネル３９０の署名に対して、Ｉｎｉｔｒｄ読み込み検証部３７１が持つカーネル３９０の署名に対する公開鍵を用いて署名検証を行う。

【００４９】

Ｓ８２０でＭＦＰ１００はカーネルを起動する。

【００５０】

Ｓ８２１でＭＦＰ１００はカーネル、Ｉｎｉｔｒｄ読み込み検証部３７１を利用して、Ｉｎｉｔｒｄ３８０の検証処理を実施し、成功したかどうか確認する。成功した場合はＳ８２２を実行し、失敗した場合はＳ８１４を実行する。検証処理としてはＦｌａｓｈＲＯＭ２１１から読み込んだ、次の起動対象であるＩｎｉｔｒｄ３８０の署名に対して、Ｉｎｉｔｒｄ読み込み検証部３７１が持つＩｎｉｔｒｄ３８０の署名に対する公開鍵を用いて署名検証を行う。

【００５１】

Ｓ８２２でＭＦＰ１００はＩｎｉｔｒｄ３８０を起動する。

【００５２】

Ｓ８２３でＭＦＰ１００は起動時検証部３８１を利用して、コントローラソフト３００の検証を実施し、成功したかどうか確認する。成功した場合はＳ８２４を実行し、失敗した場合はＳ８１４を実行する。検証処理としてはＦｌａｓｈＲＯＭ２１１から読み込んだ次の起動対象である、コントローラソフト３００に含まれる全プログラムファイルのハッシュ値をＦｌａｓｈＲＯＭ２１１から読み込んで再計算する。再計算したハッシュ値を起動時検証用正解値リスト３２１に記載されたハッシュ値とファイル毎に比較する処理が行われる。

【００５３】

Ｓ８２４でＭＦＰ１００はコントローラソフト３００の起動を開始する。コントローラソフト３００は複数のプログラムファイルに分割されているため、システムの起動のために必要なプログラムファイルが順次起動される。

【００５４】

Ｓ８２５でＭＦＰ１００は実行時検証部３２２を起動する。

【００５５】

Ｓ８１４でＭＦＰ１００は改ざんを検知したことを、操作部１０２にエラー画面８０１を表示することでユーザに通知する。

【００５６】

Ｓ８１５でＭＦＰ１００は起動シーケンスをこのステップで中止することでシステムを停止する。

【００５７】

Ｓ８２６でＭＦＰ１００はＦｌａｓｈＲＯＭ２１１に配置されたソフトウェアの検証処理を終了する。

【００５８】

一般的にソフトウェア検証はソフトウェアによって実装された検証方法であるため、記憶部のソフトウェアを書き換えることで改ざんすることができる。上記フローの様に、検証を行うソフトウェアをあらかじめ別の構成部によって検証しておくことで、改ざんされていないことを保証できる。そして、連鎖するソフトウェア検証の起点に、ハードウェア検証を用いることにより、システム全体が改ざんされていないことを保証できる。さらに、実行時検証部の起動に、ソフトウェア検証を適用させることで、システム起動後の改ざんに対しても、ハードウェア検証を起点とする強固な信頼性を確保できる。さらに、ソフトウェア更新時に実行時検証部を停止するため、プログラムの改ざんの危険性があるが、必ず再起動することによって起動時検証を行うことでＭＦＰ１００全体が改ざんされていないことを保証することができる。

【００５９】

図９を用いて、ＭＦＰ１００が機能実行時にソフトウェアを検証する処理フローを説明

する。この処理は図５に示す各機能を実行するたびに行われることを特徴とし、図８のＳ８１３以降のフローのように起動時に一度行われる処理ではない。起動時に検証しているソフトウェアであっても機能実行時に再度検証することで、起動してから機能実行までに改ざんされた場合であっても適切に検知することができる。また、機能実行時の検証だけではハードウェア検証に基づく強固な検証にならないことから、起動時検証と実行時検証を両方実施することで強固な信頼性を確保できる。図９のＭＦＰ１００が実施する処理は、ＦｌａｓｈＲＯＭ２１１に格納されたプログラムをＣＰＵ２０１がＤＲＡＭ２０２に読み込んだのち、ＣＰＵ２０１の演算処理として実施するものである。

【００６０】

ＭＦＰ１００が機能を実行する場合、必要なソフトウェアに対応するプログラムファイルの起動が必要になり、それに先立って機能実行時の検証処理を開始する。例えば、ユーザがボタン４０２を押下してコピー機能を開始すると、コピー機能の実行に必要なプログラムファイルが検証、起動される。

【００６１】

Ｓ９０１でＭＦＰ１００は実行時検証部３２２を利用して、起動対象のプログラムを検証し、検証に成功したかどうか確認する。成功した場合はＳ９０２を実行し、失敗した場合はＳ９０３を実行する。検証処理としては、実行時検証部３２２が実行時検証用ホワイトリスト３２３を参照し、起動対象プログラムに対応するハッシュ値を特定する。このハッシュ値と、ＦｌａｓｈＲＯＭ２１１に格納されたプログラムファイルから再計算したハッシュ値を比較することで、検証を行う。また、実行時検証用ホワイトリスト３２３から、起動対象プログラムのハッシュ値を特定できなかった場合は、検証失敗と判断する。

【００６２】

Ｓ９０２でＭＦＰ１００は対象プログラムを起動する。

【００６３】

Ｓ９０３でＭＦＰ１００は実行時検証部３２２を利用して、対象プログラムが既知のプログラムかどうか特定する。既知のプログラムであった場合はＳ９０５を実行し、未知のプログラムであった場合はＳ９０４を実行する。未知のプログラムであった場合とは、Ｓ９０１で起動対象プログラムのハッシュ値を特定できなかった場合であり、ソフトウェア更新部３１０による正規の手順でシステムに追加されたプログラムではないことを表している。通常、システムを改ざんさせない為に、ソフトウェア制御部３１０以外がプログラムを追加することができない構成をとるが、システムに含まれる何らかの脆弱性を利用して不正なプログラムが配置されたとしても、実行をブロックできる。この様な万が一に備えたブロック機構は、機密レベルの高い業務においてユーザが安心して利用するために必要である。既知のプログラムが改ざんされた場合、システムの制御に必要なソフトウェア部品が損なわれた状態であり、システムとしての動作を保証できない。そのため、システム全体を停止する必要がある。未知のプログラムのみが改ざんされた場合、システムの制御に必要なソフトウェア部品は全て正常な状態である。そのため、システムは停止せずに、動作を継続できる。

【００６４】

Ｓ９０４でＭＦＰ１００は操作部１０２に表示されるメッセージ表示領域４０６に、システム継続エラーとして未知のプログラムの実行をブロックした旨をユーザに通知する。この場合、メニュー画面４０１の各種ボタンは押下可能で、ＭＦＰ１００はシステムとしての動作を継続する。

【００６５】

Ｓ９０５でＭＦＰ１００は操作部１０２にエラー画面８０１を表示することでシステム停止エラーとして改ざんが行われたことをユーザに通知する。

【００６６】

Ｓ９０６でＭＦＰ１００はネットワーク制御部３０７を利用し、ネットワークからのジョブの投入を停止する。また、操作部１０２はエラー画面８０１から他の画面に遷移することはできず、ＭＦＰ１００はシステムとしての動作を停止する。

【００６７】
Ｓ９０７でＭＦＰ１００は機能実行時の検証処理を終了する。
【００６８】
以上、第１の実施形態によりプログラムの実行時に改ざんを検証するシステムにおいて、実行時検証機能が有効になっている状態でプログラム更新時に実行時検証機能もしくは書き込み制御を停止する。さらにプログラム更新終了後に再起動し起動時検証を行うことにより、セキュリティを維持しつつ速度の低下を防ぐことができる。
【００６９】
以上、第１の実施形態について述べた。
【００７０】
（第２の実施形態）
第１の実施形態では、ソフトウェア更新を行う際には必ず実行時検証機能もしくは書き込み制御を停止していた。ソフトウェア更新が部分プログラムファイルで、監視する書き込み制御の数が少ない場合は書き込み制御を停止する必要はない。
【００７１】
そこで、第２の実施形態では、更新ファイルのサイズに応じて書き込み制御を停止するかどうかを判定する。
【００７２】
図１０を用いて更新ファイルのサイズと処理時間の関係を説明する。横軸は更新ファイルのサイズＤ（Ｂｙｔｅ）、縦軸は処理時間Ｔ（秒）である。１Ｂｙｔｅ当りの書き込み時間をＴｗ（秒／Ｂｙｔｅ）とし、実行時検証機能がＯＮの場合の１Ｂｙｔｅ当りの書き込み時間をａ＊Ｔｗ（秒／Ｂｙｔｅ）（ａ＞１）とする。ＭＦＰ１００の再起動および起動時検証に要する時間をＴｂｏｏｔ（秒）とする。プログラム更新時に実行時検証機能をＯＦＦにする場合、ソフトウェア更新に要する時間は再起動も含めてＴｗ＊Ｄ＋Ｔｂｏｏｔとなる。図１０に示すように、Ｄが小さいときは実行時検証機能をＯＮのままソフトウェアを更新したほうが時間が短くて済む。ＤがＴｂｏｏｔ／［（ａ－１）Ｔｗ］より大きくなると実行時検証機能をＯＦＦにしてソフトウェアを更新し、更新後再起動したほうが時間が短くて済む。したがって、ＤとＴｂｏｏｔ／［（ａ－１）Ｔｗ］の大小関係を用いて実行時検証機能をＯＦＦにするかどうか判定を行う。ただし、実際にソフトウェアを更新する際には最初の予測よりも書き込み時間がかかる場合がありうる。そこで、書き込み中に実書き込み時間を計測し、時間がかかりそうな場合は書き込み制御をＯＦＦにするフローに戻る。
【００７３】
図１１を用いて第２の実施形態の処理フローを説明する。ただし、第１の実施形態と同じ部分は図８を用いて説明する。
【００７４】
Ｓ１１０１からＳ１１０３およびＳ１１１３はそれぞれＳ８０１からＳ８０３およびＳ８０４と同じ処理である。Ｓ１１０３で更新ファイルが存在した場合、更新用ファイルサイズＤを取得する。次のＳ１１０５でＤがＴｂｏｏｔ／［（ａ－１）Ｔｗ］より大きいか比較する。Ｙｅｓの場合はＳ１１０６を実行する。Ｎｏの場合はＳ１１１４を実行する。Ｓ１１０６では実行時検証機能がＯＮであるか確認する。もしＹｅｓであれば、Ｓ１１０７を実行する。Ｎｏであれば、Ｓ１１０８を実行する。Ｓ１１０７では、実行時検証機能の書き込み制御をＯＦＦにする。Ｓ１１０８からＳ１１１１まではそれぞれＳ８０７からＳ８１０と同じ処理である。Ｓ１１１２で書き込み制御をＯＮにする。Ｓ８１２以降は第１の実施形態と同様である。
【００７５】
Ｓ１１１４では、Ｓ８０７およびＳ１１０８と同様、更新用ファイルを展開する。Ｓ１１０５では、Ｓ８０８およびＳ１１０９と同様、起動時検証用正解値リストを反映する。Ｓ１１１６では、プログラムファイルをｄバイト書き込む。Ｓ１１１７では書き込みにかかる時間を計測し、１Ｂｙｔｅ当りの書き込み時間ＴＡを算出する。Ｓ１１１８では書き

込み制御をＯＮのまま続けた場合の残り時間の予測値ＴＡ＊（Ｄ－ｄ）が書き込み制御をＯＦＦにした場合の予測値Ｔｗ＊（Ｄ－ｄ）＋Ｔｂｏｏｔより短い場合（Ｙｅｓ）、Ｓ１１１９を実行する。長い場合（ＮＯ）、Ｓ１１２１を実行する。Ｓ１１１９では、プログラムファイルの書き込みが終了したか確認する。Ｙｅｓの場合はＳ１１２０を実行する。Ｎｏの場合はＳ１１１６に戻る。Ｓ１１２０ではＳ１１１１およびＳ８１０と同様、実行時検証用ホワイトリストを反映する。Ｓ１０２３でソフトウェア更新を終了する。Ｓ１１２１ではＳ１１０６と同様、実行時検証機能がＯＮであるか判定する。ＹＥＳの場合はＳ１１２２で書き込み制御をＯＦＦにする。Ｎｏの場合はＳ１１１０を実行する。

【００７６】

以上、第２の実施形態について述べた。本実施形態によって、更新用ファイルサイズに応じて実行時検証機能の書き込み制御をＯＮのまま更新し再起動するか、ＯＦＦにして更新し再起動しないかを、処理時間を考慮して最適に判断できる。

【００７７】

（その他の実施例）

本発明は、上述の実施形態の１以上の機能を実現するプログラムを、ネットワーク又は記憶媒体を介してシステム又は装置に供給し、そのシステム又は装置のコンピュータにおける１つ以上のプロセッサーがプログラムを読出し実行する処理でも実現可能である。また、１以上の機能を実現する回路（例えば、ＡＳＩＣ）によっても実現可能である。

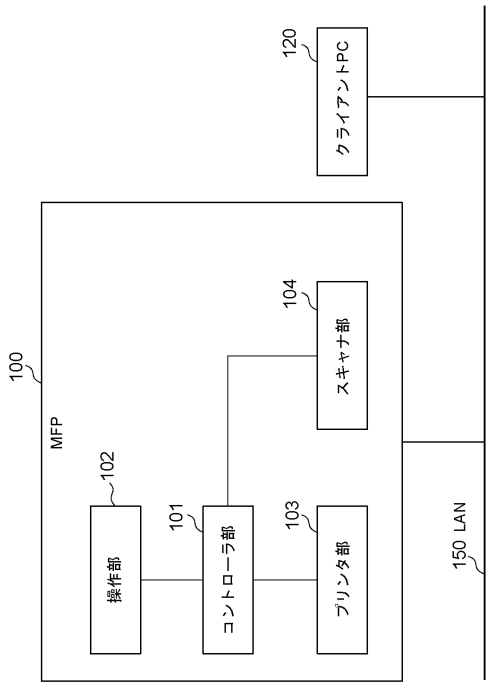
【符号の説明】

【００７８】

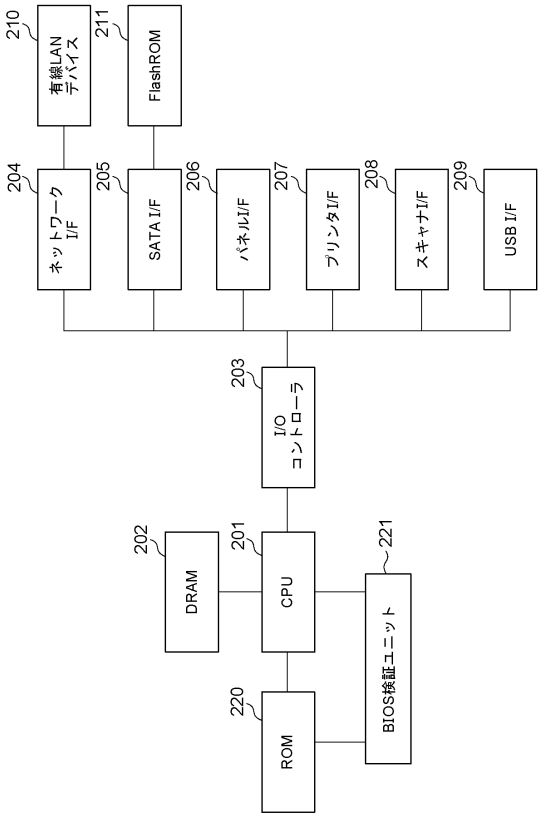
１００　ＭＦＰ
１０１　コントローラ部
２０１　ＣＰＵ
３００　コントローラソフト
２２１　ＢＩＯＳ検証ユニット
３８１　起動時検証部
３２１　起動時検証用正解値リスト
３２２　実行時検証部
３２３　実行時検証用ホワイトリスト

【図面】

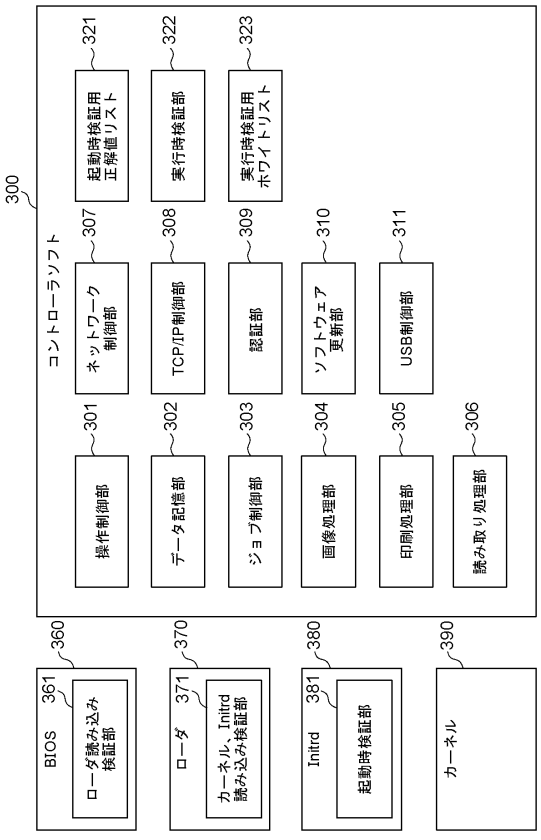
【図１】



【図２】



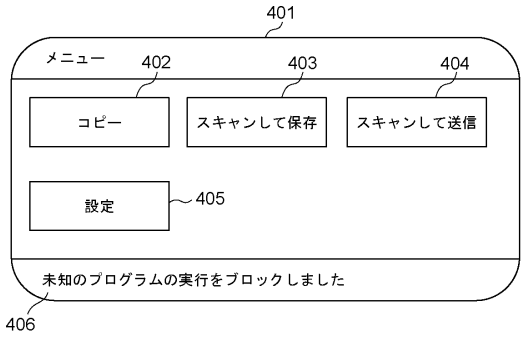
【図３】



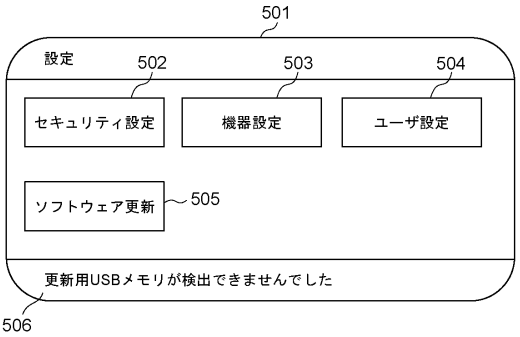
【図４】

3001 ファイル名	3002 ハッシュ
/bin/cat	270c2468bd4fdbc9e89ae71
/lib/libc.so	e26b6d7fdd32a3e50cfb5cb
/lib/libm.so	887xb9019a76dcdbbfefe25

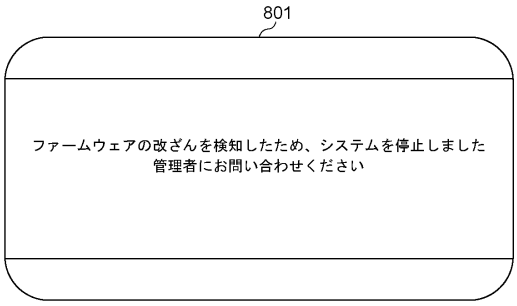
【図５】


401
メニュー
402
403
404
コピー
スキャンして保存
スキャンして送信
設定
405
未知のプログラムの実行をブロックしました
406

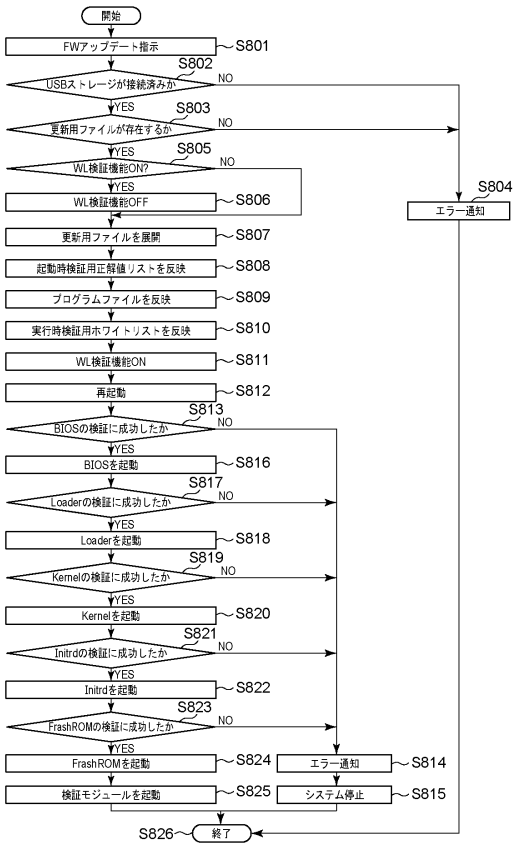
【図６】


501
設定
502
503
504
セキュリティ設定
機器設定
ユーザ設定
ソフトウェア更新
505
更新用USBメモリが検出できませんでした
506

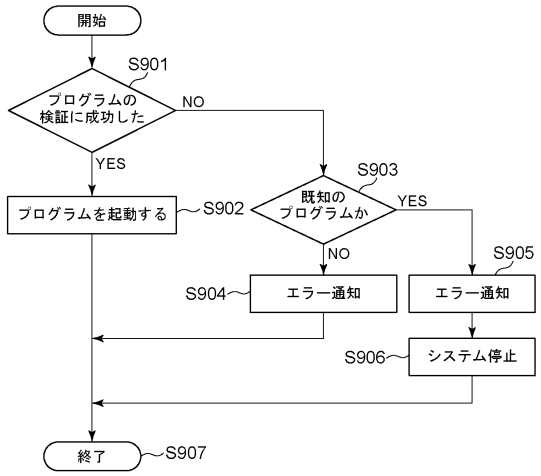
【図７】


801
ファームウェアの改ざんを検知したため、システムを停止しました
管理者にお問い合わせください

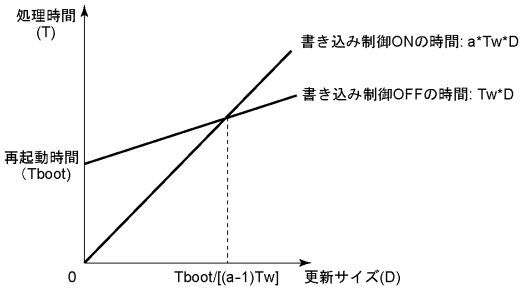
【図８】


開始
FWアップデート指示
S801
S802
USBストレージが接続済みか
NO
YES
S803
更新用ファイルが存在するか
NO
YES
S805
WL検証機能ON?
NO
YES
WL検証機能OFF
S806
S804
エラー通知
更新用ファイルを展開
S807
起動時検証用正解値リストを反映
S808
プログラムファイルを反映
S809
実行時検証用ホワイトリストを反映
S810
WL検証機能ON
S811
再起動
S812
S813
BIOSの検証に成功したか
NO
YES
BIOSを起動
S816
S817
Loaderの検証に成功したか
NO
YES
Loaderを起動
S818
S819
Kernelの検証に成功したか
NO
YES
Kernelを起動
S820
S821
Initrdの検証に成功したか
NO
YES
Initrdを起動
S822
S823
FrashROMの検証に成功したか
NO
YES
FrashROMを起動
S824
エラー通知
S814
検証モジュールを起動
S825
システム停止
S815
S826
終了

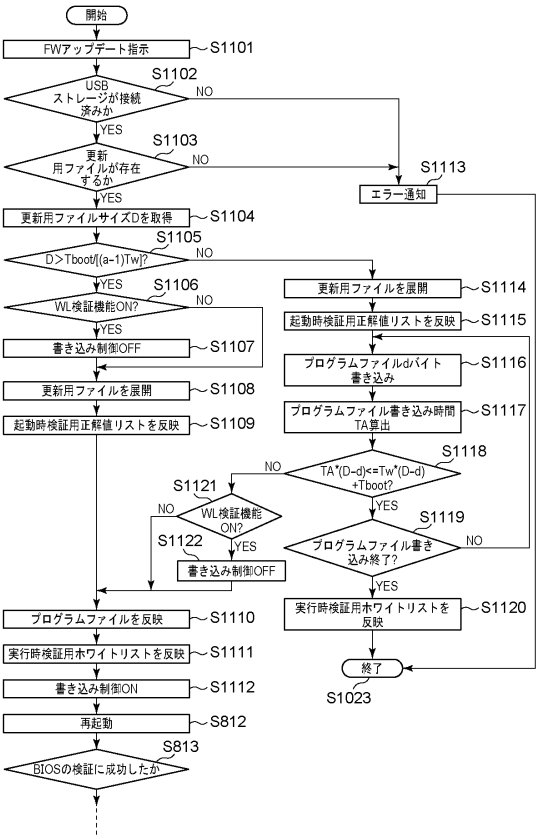
【図９】


開始
S901
プログラムの検証に成功した
NO
YES
プログラムを起動する
S902
既知のプログラムか
S903
YES
NO
S904
エラー通知
S905
エラー通知
S906
システム停止
終了
S907

【図１０】


処理時間
(T)
書き込み制御ONの時間: a*Tw*D
書き込み制御OFFの時間: Tw*D
再起動時間
（Tboot)
0
Tboot/[(a-1)Tw]
更新サイズ(D)

【図１１】


開始
FWアップデート指示
S1101
USBストレージが接続済みか
S1102
NO
YES
更新用ファイルが存在するか
S1103
NO
YES
S1113
エラー通知
更新用ファイルサイズDを取得
S1104
D>Tboot/[(a-1)Tw]?
S1105
NO
YES
WL検証機能ON?
S1106
NO
YES
書き込み制御OFF
S1107
更新用ファイルを展開
S1108
起動時検証用正解値リストを反映
S1109
更新用ファイルを展開
S1114
起動時検証用正解値リストを反映
S1115
プログラムファイルdバイト書き込み
S1116
プログラムファイル書き込み時間TA算出
S1117
TA*(D-d)<=Tw*(D-d)+Tboot?
S1118
NO
YES
WL検証機能ON?
S1121
NO
YES
書き込み制御OFF
S1122
プログラムファイル書き込み終了?
S1119
NO
YES
実行時検証用ホワイトリストを反映
S1120
プログラムファイルを反映
S1110
実行時検証用ホワイトリストを反映
S1111
書き込み制御ON
S1112
再起動
S812
BIOSの検証に成功したか
S813
終了
S1023

フロントページの続き

キヤノン株式会社内
(72)発明者　田頭 信博
東京都大田区下丸子３丁目３０番２号 キヤノン株式会社内
(72)発明者　河津 鮎太
東京都大田区下丸子３丁目３０番２号 キヤノン株式会社内

審査官　宮司 卓佳

(56)参考文献　特開２０１２－０７８９５２（ＪＰ，Ａ）
特開２００５－２０８８１４（ＪＰ，Ａ）
国際公開第２００６／１２９６５４（WO，Ａ１）
特開２０１９－０７５０００（ＪＰ，Ａ）
特開２０１２－１４６３３８（ＪＰ，Ａ）
特開２００９－１１０１３１（ＪＰ，Ａ）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　２１／５７