(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7368476号
(P7368476)

(45)発行日　令和5年10月24日(2023.10.24)

(24)登録日　令和5年10月16日(2023.10.16)

(51)国際特許分類

		FI		
G06F	*9/445(2018.01)*	G06F	9/445	
G06F	*9/455(2018.01)*	G06F	9/455	150
G06F	*21/60 (2013.01)*	G06F	21/60	320
G06F	*21/62 (2013.01)*	G06F	21/62	309
G06F	*21/53 (2013.01)*	G06F	21/53	

請求項の数　13　(全25頁)

(21)出願番号　特願2021-537931(P2021-537931)
(86)(22)出願日　令和2年1月31日(2020.1.31)
(65)公表番号　特表2022-520703(P2022-520703A)
(43)公表日　令和4年4月1日(2022.4.1)
(86)国際出願番号　PCT/IB2020/050789
(87)国際公開番号　WO2020/161577
(87)国際公開日　令和2年8月13日(2020.8.13)
審査請求日　令和4年6月22日(2022.6.22)
(31)優先権主張番号　19155755.2
(32)優先日　平成31年2月6日(2019.2.6)
(33)優先権主張国・地域又は機関
欧州特許庁(EP)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MACHINES CORPORATION
アメリカ合衆国１０５０４ ニューヨーク州 アーモンク ニュー オーチャード ロード
New Orchard Road, Armonk, New York 10504, United States of America
(74)代理人　100112690
弁理士　太佐 種一

最終頁に続く

(54)【発明の名称】　セキュア・コンテナの作成および実行

(57)【特許請求の範囲】

【請求項１】

セキュア・ソフトウェア・コンテナを作成するためのコンピュータ実施方法であって、

第１の階層化ソフトウェア・コンテナ・イメージを提供することと、

前記第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換することであり、前記ボリュームはブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む、前記変換することと、

前記レイヤの一部分のブロックの前記セットの各ブロックを暗号化することと、

前記ブロックの各暗号化セットを、前記第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックの前記セットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶することであり、

以て、セキュア暗号化ソフトウェア・コンテナが作成される、前記記憶することとを含む、コンピュータ実施方法。

【請求項２】

前記ブロックの各暗号化セットを前記記憶することはまた、

前記第１の階層化ソフトウェア・コンテナ・イメージのメタデータを記憶することも含む、請求項１に記載の方法。

【請求項３】

前記暗号化コンテナ・イメージに記憶されている前記レイヤの各々はシン・プロビジョ

ニングを適用し、シン・プロビジョニング・メタデータも含む、請求項１または２に記載の方法。
【請求項４】
前記暗号化コンテナ・イメージの前記レイヤの各ファイルの名前は、前記ファイルの内容のハッシュ値である、請求項１ないし３のいずれか一項に記載の方法。
【請求項５】
仮想機械オペレーティング・システム、開始プログラム、および復号キーを提供することも含み、前記復号キーは、ブロックの前記セットの各ブロックの前記暗号化に使用されている暗号化キーに対応する、請求項１ないし４のいずれか一項に記載の方法。
【請求項６】
前記仮想機械オペレーティング・システムによる仮想機械の開始に有効であるセキュア・コンテナ実行環境を提供することも含む、請求項５に記載の方法。
【請求項７】
前記仮想機械の前記開始は、
前記復号キーを使用して前記暗号化コンテナ・イメージのブロックのセットを復号することを含む、請求項６に記載の方法。
【請求項８】
前記第１の階層化ソフトウェア・コンテナ・イメージのレイヤのシーケンスにおいて前記第１の階層化ソフトウェア・コンテナ・イメージを再構築することも含む、請求項７に記載の方法。
【請求項９】
前記復号されているセキュア・コンテナ・イメージの上のレイヤが、リード／ライト・アクセスを可能にし、最上レイヤの下の前記レイヤが、リード・オンリー・アクセスを可能にする、請求項８に記載の方法。
【請求項１０】
前記セキュア・コンテナ実行環境は、権限のあるユーザまたは他のプロセスあるいはその両方による前記仮想機械へのアクセスを妨げるセキュア・ファームウェアによって保護され、前記仮想機械オペレーティング・システム、前記開始プログラムおよび前記復号キーが各々暗号化される、請求項６に記載の方法。
【請求項１１】
前記セキュア・ファームウェアは、ハードウェア・セキュリティ・モジュールと協働する、請求項１０に記載の方法。
【請求項１２】
セキュア・ソフトウェア・コンテナを作成するためのセキュア・コンテナ・システムであって、
第１の階層化ソフトウェア・コンテナ・イメージを受信するように適合されている受信ユニットと、
前記第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換するように適合されている変換ユニットであり、前記ボリュームはブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む、前記変換ユニットと、
前記レイヤの一部分のブロックの前記セットの各ブロックを暗号化するように適合されている暗号化モジュールと、
前記ブロックの各暗号化セットを、前記第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックの前記セットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶するように適合されている記憶ユニットであり、
以て、セキュア暗号化ソフトウェア・コンテナが作成される、前記記憶ユニットとを備える、セキュア・コンテナ・システム。
【請求項１３】

セキュア・ソフトウェア・コンテナを作成するためのコンピュータ・プログラムであって、プロセッサに、
第１の階層化ソフトウェア・コンテナ・イメージを提供することと、
前記第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換することであり、前記ボリュームはブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む、前記変換することと、
前記レイヤの一部分のブロックの前記セットの各ブロックを暗号化することと、
前記ブロックの各暗号化セットを、前記第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックの前記セットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶することであり、
以て、セキュア暗号化ソフトウェア・コンテナが作成される、前記記憶することと、
を実行させるためのコンピュータ・プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般的に、セキュア・コンピューティングに関し、より詳細には、セキュア・ソフトウェア・コンテナを作成するためのコンピュータ実施方法に関する。本発明はさらに、セキュア・ソフトウェア・コンテナを作成するための関連するセキュア・コンテナ・システム、および、コンピュータ・プログラム製品に関する。
【背景技術】
【０００２】
今日、ＩＴ（情報技術）産業においてはクラウド・コンピューティングが引き続き最も話題を集める主題となっている。個人、ならびに、小規模および中規模の会社と大企業は、大規模なクラウド・コンピューティング・センターを運営するクラウド・コンピューティング提供者にコンピューティング・タスクを外部委託し続けている。他方、ＩＴ産業における最も大きい関心事の１つが、データ・セキュリティである。したがって、私的コンピューティング環境、すなわち、企業コンピューティング・センターにおいて、セキュリティは重要な主題であり、結果、クラウド・コンピューティング環境におけるデータおよびコンピュータ・セキュリティの適合が、大企業の経営幹部レベルまで推し進められている。
【０００３】
特にＧＤＰＲ（欧州連合による一般データ保護規則）などの新たな行政機関データ・セキュリティ規則に照らして、データ・セキュリティ課題の一部は、例えばインターネットなどの公衆ネットワークを介してデータを送信している間に、または、記憶デバイスにデータを記憶するときに、あるいはその両方において、関連データを暗号化することによって対処することができる。しかしながら、データを処理する必要がある場合、データおよびプログラムは、クラウド・コンピューティング・センターの管理者によって多かれ少なかれアクセス可能であり得る。これは、データ・セキュリティ侵害につながる糸口と考えることができる。
【０００４】
一方の側におけるクラウド・コンピューティング・リソースのより高頻度の使用と、他方におけるより高いデータ・セキュリティおよびデータ・プライバシー要件との間の対立の可能性に対処するための１つの試みは、例えば、ハードウェア・セキュリティ・モジュール（ＨＳＭ）を使用した信頼できるコンピューティング環境の形態の、安全確保されたコンピューティング・プラットフォーム上で安全確保された仮想機械を使用し、厳密に境界を区切ってデータならびにアプリケーションおよび仮想機械（ＶＭ）の暗号化を可能にすることである。
【０００５】
他方、仮想機械技術も進歩しており、頻繁に使用される現行技術の１つは、各コンテナ

の各仮想機械内のオペレーティング・システムの全体的なオーバーヘッドを伴わない、オペレーティング・システムに基づくスコーピング・サンドボックス内の複数のアプリケーションと基本的に考えることができる「コンテナ・コンピューティング」である。したがって、典型的にはオペレーティング・システムおよび関連ミドルウェアによって提供される多くの一般的なサービスを、１つのオペレーティング・システム内の複数の異なるコンテナの間で共有することができる。しかしながら、この結果として、セキュリティの観点から、コンテナのデータおよびプログラムが、他のＩＴリソース（仮想機械など）と同様に管理することができなくなる可能性がある。

【０００６】

ソフトウェア・コンテナに最も頻繁に使用されるプラットフォームの１つが、Ｄｏｃｋｅｒ Ｉｎｃ．製のＤｏｃｋｅｒ ｅｎｇｉｎｅである。Ｌｉｎｕｘ環境において、これは、瞬く間にマイクロサービス・アプリケーションをパッケージするための標準となっているが、この技術はマイクロサービスのみに限定されない。基本的には、これは、分離および可搬性の特性を提供するために、容易に入手可能な商用のモノリシック・アプリケーションをパッケージするために使用することができる。

【０００７】

この文脈において、一連の刊行物が作成されている。米国特許出願公開第２０１８／０３０９７４７号明細書は、セキュリティ・モジュールと同時に作動するエージェント・エグゼクティブが、最初に実行されるときに、ユーザからエージェントＡＰＩキーを取得する、コンピュータ・システムおよび方法を開示している。このキーは、グリッド・コンピューティング・システムに通信される。一般的に、ＡＰＩキーが有効であるときの暗号トークン生成プロトコルによるエージェント識別トークンが、グリッドから受信され、エージェント・エグゼクティブと関連付けられるセキュア・データ・ストアに記憶される。エージェント・エグゼクティブの完全性を評価する情報が、エージェント自己検証ファクタを使用して収集される。

【０００８】

他方、国際公開第２０１８／００７２１３号パンフレットから、キーセットによって暗号化されているデータを記憶するセキュア記憶領域を含むＤｏｃｋｅｒイメージを安全に管理する方法が知られている。Ｄｏｃｋｅｒイメージはセキュア・ドライバを含み、方法は、以下の一連のステップを含む。すなわち、（ｉ）セキュア・ドライバに提供される入来する認証情報が目下の認証情報と一致する場合にのみ、セキュア・ドライバが信頼できるストアからキーセットを取り出す。（ｉｉ）セキュア・ドライバがセキュア記憶領域にアクセスし、キーセットを使用してデータを復号する。（ｉｉｉ）セキュア・ドライバがデータをＤｏｃｋｅｒイメージの外部に送信する。

【０００９】

しかしながら、一方におけるソフトウェア・コンテナの的確な管理と、他方におけるセキュリティ課題との間には、まだ隔たりが残っている。したがって、提案される概念の目的は、ソフトウェア・コンテナにおけるデータおよびアプリケーションのセキュリティを向上させることを可能にすることである。

【発明の概要】

【００１０】

本発明の一態様によれば、セキュア・ソフトウェア・コンテナを作成するためのコンピュータ実施方法を提供することができる。方法は、第１の階層化ソフトウェア・コンテナ・イメージを提供することと、第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換することとを含むことができる。ボリュームは、ブロックのセットを含むことができ、各レイヤは、次により低いレイヤに対する漸進的な差を含むことができる。方法は、レイヤの一部分のブロックセットの各ブロックを暗号化することと、各暗号化ブロックセットを、第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックセットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶すること

とをさらに含むことができる。それによって、セキュア暗号化ソフトウェア・コンテナを作成することができる。

【００１１】

本発明の別の態様によれば、セキュア・ソフトウェア・コンテナを作成するためのセキュア・コンテナ・システムを提供することができる。セキュア・コンテナ・システムは、第１の階層化ソフトウェア・コンテナ・イメージを受信するように適合されている受信ユニットと、第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換するように適合されている変換ユニットであって、ボリュームは、ブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む、変換ユニットと、レイヤの一部分のブロックセットの各ブロックを暗号化するように適合されている暗号化モジュールと、各暗号化ブロックセットを、第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックセットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶するように適合されている記憶ユニットとを備えることができる。したがって、セキュア・コンテナ・システムは、セキュア暗号化ソフトウェア・コンテナを作成することを可能にされ得る。

【００１２】

セキュア・ソフトウェア・コンテナを作成するための提案されているコンピュータ実施方法は、複数の利点および技術的効果を提供することができる。

【００１３】

提案される概念は、少なくとも、３つの重要なデータおよびアプリケーションのセキュリティ要件を与える。第１に、それらのメモリを含む仮想機械の内容を権限のある管理者（ハイパーバイザ管理者であっても）から隠す技法を使用することによって、コンテナ・ホスト視点およびその権限のあるシステム管理者からは、何者もコンテナのイメージの内容、すなわち、ソフトウェア・コンテナ・ファイルを見て理解することができないことが達成される。また、メモリ内で実行するコンテナへのアクセスを防止することができ、結果として、コンテナが秘密裏に、かつセキュアに実行される。

【００１４】

第２に、コンテナ・イメージおよびその管理者のいずれかを記憶するために使用されるレジストリは、コンテナ・イメージの内部で行われていることの理解から除外される。また、一般的に、制御不可能な多くの態様においてコンテナ・イメージにアクセスすることができる人のグループは、管理されているソフトウェア・コンテナの実際の内容の理解から除外され得る。例えば、それらの人々は、ファイルまたは実行可能コードまたはデータを理解し、見ることはできない。第３の中核的な利点として、階層化、使用される重複排除手法（例えば、ＣｏＷ［コピー・オン・ライト］、シン・プロビジョニング）などのようなコンテナ・イメージ属性、および、ソフトウェア・コンテナの管理の典型的な態様などの典型的な処理を保存することができることが言及され得る。コンテナの不変の管理および処理によって、使用されているコンテナ・イメージの暗号化以外に全体的なソリューションに変更を加えることなく、既存のソリューションの機密レベルを向上させることが可能であり得る。

【００１５】

したがって、提案される概念は、通常の権限を有する管理者がアクセスすることができる環境においてソフトウェア・コンテナ作業負荷を実行することを可能にし、ソフトウェア・コンテナ・イメージの内容の機密をセキュアなままにすることができる。したがって、例えば、システム管理者を信頼することができない、信頼できないコンピューティング環境も使用することができ、ソフトウェア・コンテナの作成者以外の何者も、実行中のその主記憶装置／ＲＡＭを含む、プログラム・コード、ファイルおよびデータを見ることができないことを依然として保証することができる。

【００１６】

この概念は、そのセキュリティ・キーをソフトウェア・コンテナの動作上の利点のより

多大な欠如および損失を代償にしてのみ増大し得る、仮想機械のセキュアな実行を超える。仮想機械の文脈において、セキュアな実行なしには、ソフトウェア・コンテナの内容の機密を保証することができないということが、今日までに認知されている。提案される概念は、この隔たりを埋め、単なる名前空間およびさらには保護されていない仮想機械の保護を明確に超えて、ソフトウェア・コンテナの隔離された作業空間を提供する。

【００１７】

以下において、方法および関連システムに適用可能な、本発明の概念の追加の実施形態を説明する。

【００１８】

方法の１つの好ましい実施形態によれば、各暗号化ブロックセットを記憶することはまた、第１の階層化ソフトウェア・コンテナ・イメージのメタデータを記憶することを含むこともできる。メタデータは、環境変数、コマンド、使用されるポートなどを含んでもよい。メタデータは、暗号化されてもよく、または、暗号化されなくてもよい。しかしながら、たとえメタデータが暗号化されなくとも、権限のない人間はセキュア・ソフトウェア・コンテナの実際の内容について何も言うことができない。

【００１９】

方法の１つの有利な実施形態によれば、暗号化コンテナ・イメージ内に、すなわち、ファイルとして記憶されるレイヤの各々は、シン・プロビジョニングを適用することができ、また、例えば、それに関してブロックが有効なデータを有するシン・プロビジョニング・メタデータを含むこともできる。これによって、実際に有効なアクティブ・データを保持する記憶領域のみが実際に占有される、シン・プロビジョニングの利点をもたらすことができる。

【００２０】

方法の１つの有用な実施形態によれば、暗号化コンテナ・イメージのレイヤの各ファイルの名前は、ファイルの内容のハッシュ値である。したがって、追加のメタデータは要求され得ない。ファイル上の内容のエントロピーによって、この提案されている方法ステップを使用して一意のファイル名を生成することができるとみなすことができる。

【００２１】

有利な実施形態によれば、方法はまた、仮想機械オペレーティング・システム、特にセキュア・ソフトウェア・コンテナの開始プログラムおよび復号キーを提供することを含むこともできる。復号キーは、ブロックセットの各ブロックの暗号化に使用されている暗号化キーに対応することができる。したがって、セキュア・ソフトウェア・コンテナを解凍することができ、レイヤを元のシーケンスに戻すことができ、セキュア・ソフトウェア・コンテナに含まれるアプリケーションを開始することができる。

【００２２】

別の有利な実施形態によれば、方法はまた、仮想機械オペレーティング・システムによる仮想機械の開始に有効であるセキュア・コンテナ実行環境を提供することを含むこともできる。このように、セキュア・ソフトウェア・コンテナにパッケージされているアプリケーションの実行を実際に開始させる構成要素のスタックを完成させることができる。その上、セキュア・コンテナ実行環境はまた、実質的に侵害することが不可能であり得るように、保護することもできる。

【００２３】

完全を期すために、方法の１つの好ましい実施形態によれば、仮想機械の開始は、復号キーを使用して暗号化コンテナ・イメージのブロックセットを復号することを含むことができる。セキュア・ソフトウェア・コンテナの内部のアプリケーションの所有者のみが、キーにアクセスすることができる。したがって、セキュア・ソフトウェア・コンテナの復号バージョンを提供することと、必要な環境を見ることは、一度に自動的に実施することができる。

【００２４】

１つの付加的に好ましい実施形態によれば、方法はまた、第１の階層化ソフトウェア・

コンテナ・イメージのレイヤのシーケンスにおいて第１の階層化ソフトウェア・コンテナ・イメージを再構築することを含むこともできる。非暗号化ソフトウェア・コンテナのレイヤの元の順序を示す、同様に含まれるメタデータが、この方法ステップにとって有用であり得る。

【００２５】

方法の１つの許容可能な実施形態によれば、復号されているセキュア・コンテナ・イメージの上の、すなわち、再構築されている第１の階層化ソフトウェア・コンテナ・イメージの上のレイヤが、リード／ライト・アクセスを可能にすることができ、この最上レイヤの下のレイヤが、リード・オンリー・アクセスを可能にする。したがって、シン・プロビジョニング・モデルによれば、コピー・オン・ライト・パラダイムにしたがって、書き込み動作はファイルの以前のバージョンに対する差分または差が記憶される最上レイヤにおいてのみ実施されればよい。上レイヤからすべての他のレイヤを通じて下レイヤに向かって、仮想的な意味において垂直に見ることによって、それぞれのファイルの最新バージョンを再構築するためのすべての情報が伝達される。

【００２６】

請求項５に記載の、１つのさらなる先進的で有利な方法の実施形態によれば、セキュア・コンテナ実行環境は、典型的には、ハードウェア・セキュリティ・モジュール（ＨＳＭ）、すなわち、暗号カードを使用して、権限のあるユーザまたは他のプロセスあるいはその両方による仮想機械へのアクセスを妨げるセキュア・ファームウェアによって保護され、仮想機械オペレーティング・システム、開始プログラムおよび復号キーも、各々暗号化される。

【００２７】

したがって、方法の別の有利な実施形態によれば、セキュア・ファームウェアは、ハードウェア・セキュリティ・モジュールと協働する。この技法は、仮想機械ならびに関連するソフトウェア・コンテナ内のアプリケーションおよびデータの最もセキュアな実行環境をもたらすことができる。そのようなセキュリティ・モジュールは、セキュア・ファームウェアを実行するための必須の要件である。そのようなハードウェア・ベースのデバイスがなければ、セキュア・ファームウェアも、ハイパーバイザも、仮想機械も、ソフトウェア・コンテナのアプリケーションも、実行可能ではあり得ない。

【００２８】

さらに、実施形態は、コンピュータまたは任意の命令実行システムによって、または、それらに関連して使用するためのプログラム・コードを提供するコンピュータ使用可能媒体またはコンピュータ可読媒体からアクセス可能な、関連するコンピュータ・プログラム製品の形態をとることができる。本明細書の目的のために、コンピュータ使用可能媒体またはコンピュータ可読媒体は、命令実行システム、装置、またはデバイスによって使用するための、またはそれらに関連するプログラムを記憶、通信、伝播、または輸送するための手段を含むことができる任意の装置とすることができる。

【００２９】

本発明の実施形態は、複数の異なる主題を参照して説明されていることに留意されたい。特に、いくつかの実施形態は、方法タイプの請求項を参照して説明されており、一方、他の実施形態は、装置タイプの請求項を参照して説明されている。しかしながら、当業者には、上記および以下の説明から、別途注記されない限り、１つのタイプの主題に属する特徴の任意の組合せに加えて、異なる主題に関係する特徴間の、特に、方法タイプの請求項の特徴、および装置タイプの請求項の特徴の間の任意の組合せも、本明細書内に開示されていると考えられることが推測される。

【００３０】

本発明の上記で規定された態様およびさらなる態様が、以下に記載される実施形態の例から明らかになり、実施形態の例を参照しながら説明されるが、本発明はそれらに限定されない。

【００３１】

例としてのみ、添付の図面を参照して本発明の好ましい実施形態を説明する。

【図面の簡単な説明】

【００３２】

【図１】セキュア・ソフトウェア・コンテナを作成するための本発明のコンピュータ実施方法の一実施形態のブロック図である。

【図２】非暗号化ソフトウェア・コンテナからコンテナ・ライブラリに記憶可能な暗号化ソフトウェア・コンテナへの変換プロセスのブロック図である。

【図３】セキュア・ソフトウェア・コンテナを実行するための関連する要素の完全なスタックの一実施形態のブロック図である。

【図４】セキュア・ソフトウェア・コンテナにおけるシン・プロビジョニングおよびブロックの連結を示す図である。

【図５】セキュア・ソフトウェア・コンテナを形成するためのフローチャートの第１の部分の一実施形態を示す図である。

【図６】セキュア・ソフトウェア・コンテナを形成するためのフローチャートの第２の部分の一実施形態を示す図である。

【図７】セキュア・ソフトウェア・コンテナを実行するためのフローチャートの第１の部分の一実施形態を示す図である。

【図８】セキュア・ソフトウェア・コンテナを実行するためのフローチャートの第２の部分の一実施形態を示す図である。

【図９】セキュア・コンテナ・システムの一実施形態のブロック図である。

【図１０】セキュア・コンテナ・システムを備えるコンピューティング・システムのブロック図である。

【発明を実施するための形態】

【００３３】

本明細書の文脈において、以下の慣例、用語または表現あるいはその組合せが使用され得る。

【００３４】

「セキュア・ソフトウェア・コンテナ」という用語は、権限のない人員による許可されていないアクセスに対して安全確保されているソフトウェア・コンテナを示すことができる。したがって、ソフトウェア・コンテナの内容は、権限のない人員が、ソフトウェア・コンテナの内部に記憶されているものを判定することが可能でないように、暗号化することができる。基本的に、ソフトウェア・コンテナは、アプリケーションに使用されるオペレーティング・システム・スコーピング・メカニズムによって提供されるサンドボックスと考えることができる。これらのサンドボックス内のアプリケーションは、オペレーティング・システムの残りの部分または他のコンテナ・サンドボックスからは見えない。すべてのアプリケーション・サンドボックスは同じオペレーティング・システム・インスタンス上で作動するため、コンテナ・サンドボックス内でアプリケーションを作動するときに生成されるオーバーヘッドは、これらのアプリケーションをオペレーティング・システム下のアプリケーションとして直接的に作動させるときと比較して、最小限である。具体的には、仮想化オーバーヘッドは適用されない。したがって、ソフトウェア・コンテナは、最終的には関連するメタデータとともに、単にコンテナ・サンドボックス内で実行されるアプリケーション、すなわち、１つのコンテナ・サンドボックスを別のコンテナ・サンドボックスから区別する中核的な機能と考えることができる。コンテナ技術において、コンテナ・イメージは、コンテナ・インスタンスに提示されるすべてのファイルを含む、アプリケーションの変更不可能なパッケージングを表す。これは、コンテナ・インスタンスの間で共有することができ、また、他のコンテナ・ホストと交換することもできるアプリケーションの「青写真」である。コンテナ・インスタンスは、コンテナ・イメージをその（ルート）ファイル・システムの開始点として使用する、作動しているコンテナである。現在、Ｄｏｃｋｅｒが第一級のソフトウェア・コンテナ技術であると考えることができる。

【００３５】

セキュア・ソフトウェア・コンテナのすべてのレイヤを暗号化する必要があり得るとは限らないことも留意され得る。例えばオペレーティング・システム（例えばＵｂｕｎｔｕ）の構成要素など、共通のソフトウェア構成要素を含むレイヤは暗号化されないままにすることができる。これらの構成要素は、複数のコンテナにとって共通であり得、結果、それらの機能はいずれにせよ知ることができるため、暗号化は不要とすることができる。しかしながら、中核的なアプリケーション・ブロックおよび関連するデータは、ソフトウェア・コンテナの特徴的な構成要素と考えることができる。したがって、これらの特徴的な構成要素は、セキュア・ソフトウェア・コンテナの一部として暗号化することができる。しかしながら、完全を期すために、無論、ソフトウェア・コンテナのすべてのレイヤが暗号化されてもよいことが留意され得る。複数のソフトウェア・コンテナに共通であり得る部分を、コンテナの中核的な部分、すなわち、アプリケーションと比較して別の暗号化方法（例えば別のキーのみ、または、まったく異なる暗号化方法も）を用いて暗号化することも可能であり得る。

【００３６】

「第１の階層化ソフトウェア・コンテナ・イメージ」という用語は、それによってソフトウェア・コンテナを定義することができるレイヤのスタックを示すことができる。複数の異なるレイヤに使用されるストレージ技術は、シン・プロビジョニング／スペア・ストレージに基づくことができる。アプリケーションは最上レイヤへのライト・アクセス（およびリード・アクセス）のみが可能であり得る。最上レイヤの下のレイヤはリード・オンリーであり得る。

【００３７】

「メタデータ」という用語は「データに関するデータ」を示すことができる。ここで、メタデータは、ブロックに基づくセキュア・ソフトウェア・コンテナを復号した後に初期ソフトウェア・コンテナを再構築するための、ソフトウェア・コンテナの複数の異なるレイヤに関する、特に複数の異なるレイヤの正確な順序に関する情報を含むこともできる。

【００３８】

「ボリューム」という用語は、ここでは、仮想記憶デバイスの形態の記憶システムを示すことができる。これは、データ・ブロックの集合によって構成される。ボリュームが仮想である場合、そのデータのすべてが、その仮想ボリュームとして機能するファイルによって支援される。シン・プロビジョニング・メカニズムを使用するとき、それに従って未使用データ・ブロックをボリューム内でマークすることができ、その結果として、仮想ボリュームにサービスするファイルの空間消費を低減することができる。

【００３９】

「ブロックセット」という用語は、例えば、記憶デバイスの連結されているブロックの連続的なシーケンスなど、記憶ブロックのグループを示すことができる。

【００４０】

「漸進的な差」という用語は、セキュア・ソフトウェア・コンテナの２つの異なるレイヤの間の差分を示すことができる。この概念は、典型的には、シン・プロビジョニング概念において使用され得る。複数の異なるレイヤから成るシーケンスを新たなデータまたは更新されたデータあるいはその両方のシーケンスに反映することができる場合、情報全体（アプリケーションおよび関連するデータ）を再構築することができる。漸進的な差の概念を使用することによって、実際に必要な記憶容量のみを使用すること、すなわち、シン・プロビジョニングが可能になるが、レイヤあたりのブロックは実質的にはるかにより大きい空間（特定のレイヤまたはボリュームの記憶限界まで）を提供することができる。

【００４１】

「レイヤの部分」という用語は、ソフトウェア・コンテナのレイヤの総数よりも小さいものであり得るいくつかの数のレイヤを示すことができる。上記で言及したように、ソフトウェア・コンテナのすべてのレイヤを暗号化することは不要であり得る。標準的なソフトウェア構成要素（例えば、オペレーティング・システムの一部）を含むいくつかのより低いレイヤは、暗号化されないままとすることができる。しかしながら、他の実施形態に

おいて、これらのレイヤもまた暗号化されてもよい。

【００４２】

「暗号化コンテナ・イメージ」という用語は、暗号化形態の第１の階層化ソフトウェア・コンテナ・イメージの情報を含むことができるセキュア・ソフトウェア・コンテナのレイヤのシーケンスを示すことができる。

【００４３】

「ブロックセットの順序」という用語は、セキュア・ソフトウェア・コンテナの複数の異なるレイヤを再構築するためのブロックのシーケンスを示すことができる。

【００４４】

「セキュア暗号化ソフトウェア・コンテナ」という用語は、ここでは、セキュア・ソフトウェア・コンテナを示すことができる。セキュア・ソフトウェア・コンテナが暗号化レイヤを含むと言う事実は、本明細書内では当然のことと考えることができる。

【００４５】

「シン・プロビジョニング」という用語は、実際に利用可能であるよりも多くの物理リソースを有するように見えるように仮想化技術を使用することを示すことができる。システムが仮想化リソースを常に最大限に支援する場合、すべての容量が最初に提供されなければならない。シン・プロビジョニングという用語は、ディスク・レイヤに適用することができるが、任意のリソースの配分方式を参照してもよい。例えば、コンピュータ内の現実のメモリは、典型的には、仮想化を行う何らかの形態のアドレス変換技術によってタスクを実行するためにシン・プロビジョニングすることができる。各タスクは、現実のメモリが配分されているかのように機能することができる。配分される仮想メモリの総和を、典型的には現実のメモリの合計を超えることができるタスクに割り当てることができる。これは、レイヤおよびソフトウェア・コンテナに適用することができる。シン・プロビジョニングはまた、文脈によっては、「スパース・ボリューム」と呼ばれる場合もある。

【００４６】

「ハッシュ値」という用語は、任意のサイズのデータを固定サイズのデータにマッピングするのに使用することができるハッシュ関数の結果を示すことができる。ハッシュ関数によって返される値は、ハッシュ値、ハッシュ・コード、ダイジェスト、または単純にハッシュと呼ばれる。ハッシュ関数は、迅速なデータ検索のためにコンピュータ・ソフトウェアにおいて使用される一般的なデータ構造であるハッシュ・テーブルと組み合わせて使用されることが多い。ハッシュ関数は、大きいファイル内の重複するレコードを検出することによってテーブルまたはデータベースの検索を加速させる。１つのそのような用途は、ＤＮＡ配列内の類似の区間を発見することである。ハッシュ値の計算は一方向的な手法であるため、ハッシュ値から元の値を一意に判定することはできない。しかしながら、この技法は、暗号学において広く使用されている。

【００４７】

「仮想機械オペレーティング・システム」という用語は、ハイパーバイザ上で実行されている仮想サーバの仮想機械の構成要素であるオペレーティング・システムを示すことができる。異なる仮想機械オペレーティング・システムは、他の仮想機械オペレーティング・システムの機能に影響を及ぼし得ない。この技術は、Ｌｉｎｕｘに基づくオペレーティング・システムに広く使用することができる。しかしながら、メインフレームおよびミッドレンジのオペレーティング・システムも、この技術を使用することができる。

【００４８】

「開始プログラム」という用語は、セキュア・ソフトウェア・コンテナの実行開始を始動することを可能にされているプログラムを示すことができる。開始プログラムはまた、復号化プロセスの始動、および、第１の階層化ソフトウェア・コンテナの元のレイヤを再構築するための復号されているブロックの並べ替えを引き継ぐこともできる。

【００４９】

「復号キー」という用語は、いかなる復号機能もなしに読み出すことができるように、暗号化コンテキストをその元のコンテキストにリセットするのに有用なソフトウェア構成

要素を示すことができる。したがって、これはクリア・テキストであり得る。
【００５０】
「セキュア・コンテナ実行環境」という用語は、セキュア暗号化ソフトウェア・コンテナ、または同じくセキュア暗号化ソフトウェア・コンテナ・イメージを受け入れ、それに基づいてコンテナ・インスタンスを作動させることができるソフトウェア構成要素を示すことができる。コンテナを、ちょうどこのコンテナを作動させるように作成されている仮想機械において作動させることを選択することができる。仮想機械は、仮想機械を始動するとハイパーバイザにもなるセキュア・コンテナ実行環境の権限のある管理者が、コンテナを作動させている仮想機械の内部のデータにアクセスすることができなくなるように、安全確保することができる。セキュア暗号化ソフトウェア・コンテナ（イメージ）の復号を可能にするために、（仮想機械）オペレーティング・システムに提供することができる復号キーを使用し、コンテナ・インスタンスの立ち上げ中にプログラムを開始することができる。
【００５１】
「ハードウェア・セキュリティ・モジュール」（ＨＳＭ）という用語は、強力な認証のためのデジタル・キーを保護および管理し、暗号処理を提供することができる物理コンピューティング・デバイスを示すことができる。これらのモジュールは従来通り、コンピュータもしくはネットワーク・サーバ、または直接的にＣＰＵに取り付けられるプラグイン・カードまたは外部デバイスの形態で実現することができる。
【００５２】
「Ｄｏｃｋｅｒコンテナ」という用語は、コンテナ化としても知られる、オペレーティング・システム・レベルの仮想化の一例を示すことができる。これは、カーネルが複数の分離されたユーザ空間インスタンスの存在を可能にすることができるオペレーティング・システム機能を示すことができる。（ソフトウェア）コンテナ、パーティション、仮想環境（ＶＥ）またはジェイル（ＦｒｅｅＢＳＤジェイルまたはｃｈｒｏｏｔジェイル）と呼ばれる、そのようなインスタンスは、それらの中で作動しているプログラムの視点からは現実のコンピュータのように見え得る。通常のオペレーティング・システム上で作動しているコンピュータ・プログラムは、そのコンピュータのすべてのリソース（接続されているデバイス、ファイルおよびフォルダ、ネットワーク・シェア、ＣＰＵ電力、量子化可能なハードウェア能力）を見ることができる。しかしながら、コンテナ内部で作動しているプログラムは、コンテナの内容およびコンテナに割り当てられているデバイスしか見ることができない。したがって、コンテナは、仮想機械の互いに対する分離と同様に、互いに対して分離することができる。ここで提案される概念は、有利にはＤｏｃｋｅｒコンテナによって実施することができる。
【００５３】
以下に、図面の詳細な説明を与える。図面内のすべての指示は概略である。最初に、セキュア・ソフトウェア・コンテナを作成するための本発明のコンピュータ実施方法の一実施形態のブロック図が与えられる。その後、さらなる実施形態、および、セキュア・ソフトウェア・コンテナを作成するためのセキュア・コンテナ・システムの実施形態を説明する。
【００５４】
図１は、例えば、セキュアＤｏｃｋｅｒコンテナなど、セキュア・ソフトウェア・コンテナを作成するためのコンピュータ実施方法１００の一実施形態のブロック図を示す。方法は、特に「差分に基づく」、すなわち、シン・プロビジョニングに基づくレイヤを含むイメージなど、第１の階層化ソフトウェア・コンテナ・イメージを提供すること１０２を含むことができる。
【００５５】
方法１００は、さらに、第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルを、特にファイル・システムを含むボリュームなどのボリュームに変換すること１０４を含む。ボリュームは、特に記憶ディスクに

よって管理されるブロックなど、ブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む。メタデータは特に、コンテナに対処する方法、すなわち、複数の異なるレイヤがどのように互いに関係するかを説明する、すべての必要なメタデータであることも言及され得る。

【００５６】

加えて、方法１００は、レイヤの一部分のブロックセットの各ブロックを暗号化すること１０６を含む。これは、安全確保されたコンテナの作成者のみが知る暗号化キーを使用することによって実施することができる。すべてのレイヤを暗号化する必要があるとは限らないことも留意され得る。特に、すべてのソフトウェア・コンテナについて同一であるレイヤは、暗号化しなくてもよい。一例は、例えば「Ｕｂｕｎｔｕレイヤ」などの基本オペレーティング・システム・レイヤであり得る。

【００５７】

さらに、方法１００は、各暗号化ブロックセットを、第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックセットの順序を再構築するための非暗号化メタデータ（例えば、親レイヤへのポインタ）とともに、特にファイルとして表される暗号化コンテナ・イメージのレイヤとして記憶すること１０８を含む。この結果として、元のソフトウェア・コンテナ、すなわち、第１の階層化ソフトウェア・コンテナ・イメージのレイヤの数と同じ数のブロック・デバイスがもたらされる。それによって、セキュア暗号化階層化ソフトウェア・コンテナが作成される。

【００５８】

図２は、非暗号化ソフトウェア・コンテナ２０２からコンテナ・ライブラリに記憶可能な暗号化ソフトウェア・コンテナへの変換プロセスのブロック図２００を示す。個々のレイヤ２０４としても示されている第１の階層化ソフトウェア・コンテナ・イメージ２０２は、同じく複数の異なるレイヤ２１０を含むセキュア・ソフトウェア・コンテナ２０８に変換される。変換は、変換ツール２０６を使用して実施され、湾曲した矢印２０６ａによって示される。ソフトウェア・コンテナ２０２は暗号化されておらず、一方、セキュア・ソフトウェア・コンテナ２０８は暗号化レイヤ２１０を有する。これらは、コンテナ・リポジトリ２１２内に記憶することができる。またここで、レイヤ２１０を有するセキュア・ソフトウェア・コンテナ２０８も示されており、一例は、記号形態のレイヤ２１０を有するものである。

【００５９】

図３は、セキュア・ソフトウェア・コンテナ２０８を実行するための関連する要素の完全なスタック３００の一実施形態のブロック図を示す。仮想機械３０２のセキュア実行プラットフォーム、または代替的に、仮想機械においてコンテナを作動させるセキュア・コンテナ実行環境に基づいて、コンテナ・エンジン３０４が、ソフトウェア・コンテナを動作させるための基本要件を管理する。完全を期すために、セキュア・ソフトウェア・コンテナ２０８（明示的には示されていない）が、特に最上のリード／ライト・レイヤなどのそのレイヤ２１０とともに、暗号化形態で示されている。このコンテナ・エンジン３０４は、ハイパーバイザ３０８を含むコンテナ・ランタイム環境３０６を呼び出す。その後、ハイパーバイザ３０８は、ソフトウェア・コンテナ２０８のレイヤ内の、特にオペレーティング・システム・カーネルなどのカーネル３１０を開始する。矢印３１２は、レイヤ２１０を有するセキュア・ソフトウェア・コンテナ２０８のコンテナ・エンジンから仮想機械３１４への動きを示すことができ、仮想機械において、これは復号されて、実行可能な復号された（ｄ）コンテナ・レイヤ・スタック２０４（ｄ）（明示的には示されていない）の復号された形態のレイヤ２０４になる。

【００６０】

ハードウェア・セキュリティ・モジュールの使用、または、例えばルート・ユーザなどの権限のあるハイパーバイザ管理者が仮想機械のデータにアクセスし得ないように仮想機械を作動させるための任意の他の技術などの、仮想機械のセキュアな実行のための技術を適用することができる場合、仮想機械３１４は信頼できる環境であることが留意され得る。

【００６１】
図４は、セキュア・ソフトウェア・コンテナにおけるシン・プロビジョニングおよびブロックの連結を示す。左上側のレイヤ４０２、４０４、４０６、４０８（図４（ａ））は、ソフトウェア・コンテナ・イメージのレイヤのシーケンスであり得る。最下レイヤ４０２は、ファイル４１０Ａを含み、次に上側のレイヤ４０４は、ファイル４１２Ｂを含む。ファイル４１０Ａが更新（変更）された場合、これは、ファイル４１４Ａ’として新たなレイヤ４０６に書き込まれる。この時点において、レイヤ４０６がレイヤ４０６へのリード、および、特にライト・アクセスを可能にする最上レイヤであり得る。ソフトウェア・コンテナの完全な元の内容は、最上レイヤ４０６を介して始まってレイヤ４１４を介してレイヤ４０２へと進む、複数の異なるレイヤを通じた垂直視を基本的に表す４０８において認識することができる。
【００６２】
右上側（図４（ｂ））において、スペア・ブロック・デバイス４１６、４１８、４２０、４２２へのマッピングが示されている。複数の異なるレイヤに示されている黒色ボックスは、スペア・ブロック・デバイス内のレイヤ４０２、４０４、４０６のデータを担持するブロックを表すことができ、クライアント・キーによって暗号化することができる。したがって、図４（ｂ）のスタックは、セキュア・ソフトウェア・コンテナのデータを表すことができる。すべての暗号化ブロック・デバイス・ファイルは、ファイルとして対応する新たなコンテナ・レイヤに書き込まれる。これらのレイヤは、３つのファイルを含むことができる。それらの自動的に生成される名前は、そのファイル、すなわち、（ｉ）内容ファイル（ブロックごとの暗号化スペア・ブロック・ループバック・デバイス・ファイル）、（ｉｉ）先行するレイヤのハッシュ（すなわち、先行するレイヤのファイル名を参照するａｐｅ）のみを含む親ファイル、（ｉｉｉ）環境変数、ポート設定などのような、関連するレイヤに追加されるコンテナ・イメージ・メタデータを含むメタデータ・ファイル（これは任意選択であってもよい）の対応する内容のハッシュに基づく。この最後のファイルはまた、任意選択的に暗号化されてもよい。
【００６３】
すべてのレイヤが重なり合ってマウントされる場合、すべてのファイルが見えることになる。総計親ファイアウォールは、ブロック・デバイス・ファイルを正しい順序において互いに重ね合わせてマウントすることを可能にするレイヤの依存関係グラフを記述する。これは、図４（ｃ）に示されている。湾曲した矢印は、それぞれの親レイヤのハッシュ・アドレスの指示機能を示す。
【００６４】
図５は、セキュア・ソフトウェア・コンテナを形成するためのフローチャート５００の第１の部分の一実施形態を示す。最初に、スペア・ループバック・デバイス・ファイルが作成される５０２。次いで、ループバック・デバイス・ファイル内にファイル・システムが作成される５０４。ソフトウェア・コンテナのレイヤの底部から開始して、第１の（元の）階層化ソフトウェア構成要素のイメージの次のレイヤ「Ｌ」が識別され、その後、スペア・ループバック・デバイス・ファイル「Ｄ」が作成される５０８。デバイス・マッパが使用されて、ループバック・デバイス・ファイルの先行するスタックの上にループバック・デバイス・ファイル「Ｄ」が追加される５１０。したがって、書き込み動作は「Ｄ」に進み、要求されるオフセットにおいてブロックが見つかるまで、レイヤのスタックを通じてすべての読み出し動作がこれを要求する（５１２参照）。
【００６５】
次いで、マウント点におけるループバック・デバイス・ファイルのスタックに対するマウント動作が実施される５１４。次に、「Ｌ」のすべての内容がマウント点（これらの変更を「Ｄ」に書き込む）に追加される５１６。最後ではあるが最も重要なこととして、マウント点がアンマウントされる５１８。ステップ５０６～５１８は、すべてのイメージ・レイヤについて実施される。
【００６６】

図６は、セキュア・ソフトウェア・コンテナを形成するための図５のフローチャート５００の第２の部分６００の一実施形態を示す。フローチャートは、「Ｄ」のスペア・ループバック・デバイスを識別すること６０２によって継続し、それによって、最下レイヤから開始する。次いで、「Ｄ」が暗号化され、「Ｄ」の暗号化バージョン「Ｅ」が作成される６０４。これは、フローチャートの左側のループバック矢印によって示されるように、すべてのイメージ・レイヤについて行われる。

【００６７】

次いで、スペア・ループバック・デバイス・ファイル「Ｄ」が識別され、ここでもまた、プロセスは最下レイヤから開始する（６０６）。次に、「Ｅ」のハッシュ値「Ｈ」が作成される６０８。次のステップにおいて、再び、最下レイヤから開始して、元のイメージ・レイヤ「Ｌ」による識別が実施される６１０。６１２において、新たなコンテナ・イメージ・レイヤ「Ｍ」が任意の既存の新たなコンテナ・イメージ・レイヤの上に作成される。次いで、「Ｅ」の内容を「Ｈ」内容と呼ばれる新たなファイルに入れることによって、「Ｍ」内のファイルが作成される６１４。また、「Ｌ」の任意のメタデータ情報を新規ファイル「Ｈ」メタデータに入れることによって、「Ｍ」内のファイルが作成される６１６。

【００６８】

次に、「Ｌ」が最下レイヤであるか否かが判定される６１８。そうでない場合（「Ｎ」）、先行するハッシュ値（すなわち、先行する「Ｅ」の先行する「Ｈ」）を「Ｈ」親と呼ばれる新たなファイルに入れることによって、「Ｍ」内のファイルが作成される６２０。判定６１８が真である場合、すなわち「Ｙ」の場合、プロセスは、スパース・ループバック・デバイス・ファイル「Ｄ」を識別するステップ６０６へとループバックし、すべてのイメージ・レイヤについてこのループを繰り返す。

【００６９】

図７は、セキュア・ソフトウェア・コンテナを実行するためのフローチャート７００の第１の部分の一実施形態を示す。最初に、ホストのコンテナ・エンジンが、すべてのレイヤ「Ｅ」を互いに重ね合わせてマウントして、コンテナ・ファイル・システムを組み立てる７０２。次のステップにおいて、ホスト内の内容エンジンが、空のリード／ライトをコンテナ・ファイル・システムにマウントする７０４。次いで、コンテナ・ファイル・システムが、「セキュア実行」仮想機械を作成し７０６、その後、ファイル・システムが仮想機械に提供される７０８。コンテナ・エンジンが、ユーザによって提供されるカーネル、および、ｉｎｉｔｒｄ、ｉｎｉｔなどのコマンドならびにクライアント・キーを読み出して、暗号化ソフトウェア・コンテナ、すなわち、その内容を復号する７１０。

【００７０】

ＶＭが、ユーザによって提供されるカーネル／ｉｎｉｔｒｄ（「ｉｎｉｔｒｄ」はブート・プロセス中にＬｉｎｕｘカーネルによって使用される一時ファイル・システムである初期ＲＡＭディスクを表す）をブートし７１２、ユーザによって提供されるｉｎｉｔプロセス、すなわち、始動プロセス／コマンドを開始する７１４。次いで、フローチャートは図８において継続する。

【００７１】

図８は、セキュア・ソフトウェア・コンテナを実行するためのフローチャート７００（図７参照）の第２の部分８００の一実施形態を示す。ｉｎｉｔプロセスが、「Ｈ」親ファイルに基づいてレイヤの順序を作成し直し８０２、ｉｎｉｔプロセスが、ユーザによって提供されるクライアント・キーを用いて、すべての「Ｈ」内容ファイルをオン・ザ・フライで（例えば、「ｄｍ－ｃｒｙｐｔ」、すなわち、Ｌｉｎｕｘカーネルのデバイス・マッパの暗号モジュールを使用して）復号する８０４。さらに、ｉｎｉｔプロセスが、復号されているスパース・ループバック・デバイスを正しい順序で組み立て８０６、リード・オンリーであるすべての他のレイヤの上でリード／ライト・スパース・ループバック・デバイス・ファイルを組み立てる８０８。

【００７２】

加えて、ｉｎｉｔプロセスは、再びクライアントによって提供される復号キーを使用し

て、リード／ライト・ループバック・ファイルを新たな「H」内容ファイルとして暗号化する８１０。最後ではあるが最も重要なこととして、ｉｎｉｔプロセスは、最上リード・オンリー・レイヤを参照する新たな「H」親ファイルを暗号化する８１２。

【００７３】

完全を期すために、図９は、セキュア・コンテナ・システム９００の一実施形態のブロック図を示す。システム９００は、第１の階層化ソフトウェア・コンテナ・イメージを受信するように適合されている受信ユニット９０２と、第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルを、ブロックのセットを含むボリュームに変換するように適合されている変換ユニット９０４であって、各レイヤは、次により低いレイヤに対する漸進的な差を含む、変換ユニット９０４とを備える。

【００７４】

加えて、システム９００は、レイヤの一部分のブロックセットの各ブロックを暗号化するように適合されている暗号化モジュール９０６と、各暗号化ブロックセットを、第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックセットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶するように適合されている記憶ユニット９０８とを備える。それによって、セキュア暗号化ソフトウェア・コンテナが作成される。

【００７５】

本発明の実施形態は、プログラム・コードを記憶または実行しあるいはその両方を行うのに適しているプラットフォームにかかわらず、実質的に任意のタイプのコンピュータとともに実施することができる。図１０は、一例として、提案されている方法に関係するプログラム・コードを実行するのに適したコンピューティング・システム１０００を示す。

【００７６】

コンピューティング・システム１０００は、適切なコンピュータ・システムの一例に過ぎず、コンピュータ・システム１０００が上記に記載されている機能のいずれかを実装または実施あるいはその両方を行うことが可能であるか否かにかかわらず、本明細書において説明されている本発明の実施形態の使用または機能の範囲に関するいかなる限定を示唆するようにも意図されるものではない。コンピュータ・システム１０００内には、多数の他の汎用または専用コンピューティング・システム環境または構成を用いて動作可能である構成要素が存在する。コンピュータ・システム／サーバ１０００による使用に適することができる既知のコンピューティング・システム、環境、または構成あるいはその組合せの例は、パーソナル・コンピュータ・システム、サーバ・コンピュータ・システム、シン・クライアント、シック・クライアント、ハンドヘルドまたはラップトップ・デバイス、マルチプロセッサ・システム、マイクロプロセッサベースのシステム、セット・トップ・ボックス、プログラム可能家電製品、ネットワークＰＣ、ミニコンピュータ・システム、メインフレーム・コンピュータ・システムおよび上記のシステムまたはデバイスのいずれかを含む分散クラウド・コンピューティング環境などを含むが、これらには限定されない。コンピュータ・システム／サーバ１０００は、コンピュータ・システム１０００によって実行されている、プログラム・モジュールのようなコンピュータ・システム実行可能命令の一般的な文脈において説明され得る。一般的に、プログラム・モジュールは、特定のタスクを実行するかまたは特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、論理、データ構造などを含み得る。コンピュータ・システム／サーバ１０００は、タスクが、通信ネットワークを通じてリンクされる遠隔処理デバイスによって実行される、分散クラウド・コンピューティング環境において実践されることもできる。分散クラウド・コンピューティング環境において、プログラム・モジュールは、メモリ記憶デバイスを含むローカルおよび遠隔コンピュータ・システム記憶媒体の両方内に位置することができる。

【００７７】

図面に示すように、コンピュータ・システム／サーバ１０００は、汎用コンピューティ

ング・デバイスの形態で示されている。コンピュータ・システム／サーバ１０００の構成要素は、限定ではないが、１つまたは複数のプロセッサまたは処理ユニット１００２、システム・メモリ１００４、および、システム・メモリ１００４を含む様々なシステム構成要素をプロセッサ１００２に結合するバス１００６を含むことができる。バス１００６は、メモリ・バスまたはメモリ・コントローラ、周辺バス、アクセラレイテッド・グラフィックス・ポート、および、様々なバス・アーキテクチャのうちのいずれかを使用するプロセッサまたはローカル・バスを含む、いくつかのタイプのバス構造のいずれかのうちの１つまたは複数を表す。例として、限定ではなく、このようなアーキテクチャは、業界標準アーキテクチャ（ＩＳＡ）バス、マイクロ・チャネル・アーキテクチャ（ＭＣＡ）バス、拡張ＩＳＡ（ＥＩＳＡ）バス、ビデオ電子機器標準規格化協会（ＶＥＳＡ）ローカル・バス、および周辺構成要素相互接続（ＰＣＩ）バスを含む。コンピュータ・システム／サーバ１０００は、典型的には、様々なコンピュータ・システム可読媒体を含む。そのような可読媒体は、コンピュータ・システム／サーバ１０００によってアクセス可能である任意の利用可能な媒体であってもよく、揮発性および不揮発性両方の媒体、取り外し可能および固定媒体を含む。

【００７８】

システム・メモリ１００４は、ランダム・アクセス・メモリ（ＲＡＭ）１００８またはキャッシュ・メモリ１０１０あるいはその両方などの揮発性メモリの形態のコンピュータ・システム可読媒体を含むことができる。コンピュータ・システム／サーバ１０００は、他の取り外し可能／固定、揮発性／不揮発性コンピュータ・システム記憶媒体をさらに含んでもよい。例としてのみ、固定不揮発性磁気媒体（図示せず、典型的には「ハード・ドライブ」と呼ばれる）に対して読み書きするためのストレージ・システム１０１２を提供することができる。図示されていないが、取り外し可能不揮発性磁気ディスク（例えば、「フロッピー（Ｒ）ディスク」）に対して読み書きするための磁気ディスク・ドライブ、および、ＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭまたは他の光媒体などの取り外し可能不揮発性光ディスクに対して読み書きするための光ディスク・ドライブを提供することができる。そのような場合、各々を、１つまたは複数のデータ・メディア・インターフェースによってバス１００６に接続することができる。下記にさらに示し、説明するように、メモリ１００４は、本発明の実施形態の機能を実行するように構成されているプログラム・モジュールのセット（例えば、少なくとも１つのプログラム・モジュール）を有する少なくとも１つのプログラム製品を含むことができる。

【００７９】

限定ではなく例として、プログラム・モジュール１０１６のセット（少なくとも１つのプログラム・モジュール）を有するプログラム／ユーティリティ、ならびに、オペレーティング・システム、１つまたは複数のアプリケーション・プログラム、他のプログラム・モジュール、およびプログラム・データを、メモリ１００４に記憶することができる。オペレーティング・システム、１つまたは複数のアプリケーション・プログラム、他のプログラム・モジュール、およびプログラム・データの各々またはそれらの何らかの組合せは、ネットワーキング環境の実施態様を含むことができる。プログラム・モジュール１０１６は、一般的に、本明細書に記載されているような本発明の実施形態の機能または方法論あるいはその両方を実行する。

【００８０】

コンピュータ・システム／サーバ１０００はまた、キーボード、ポインティング・デバイス、ディスプレイ１０２０など、ユーザがコンピュータ・システム／サーバ１０００と対話することを可能にする１つまたは複数のデバイス、または、コンピュータ・システム／サーバ１０００が１つまたは複数の他のコンピューティング・デバイスと通信することを可能にする任意のデバイス（例えば、ネットワーク・カード、モデムなど）、あるいはその組合せなどの、１つまたは複数の外部デバイス１０１８と通信することもできる。そのような通信は、入出力（Ｉ／Ｏ）インターフェース１０１４を介して行うことができる。またさらに、コンピュータ・システム／サーバ１０００は、ネットワーク・アダプタ１

０２２を介して、ローカル・エリア・ネットワーク（ＬＡＮ）、一般的な広域ネットワーク（ＷＡＮ）、または公衆ネットワーク（例えば、インターネット）あるいはその組合せなどの１つまたは複数のネットワークと通信することができる。図示のように、ネットワーク・アダプタ１０２２は、バス１００６を介してコンピュータ・システム／サーバ１０００の他の構成要素と通信することができる。図示されていないが、他のハードウェア構成要素またはソフトウェア構成要素あるいはその両方が、コンピュータ・システム／サーバ１０００とともに使用されてもよいことは理解されたい。例は、限定ではないが、マイクロコード、デバイス・ドライバ、冗長処理ユニット、外部ディスク・ドライブ・アレイ、ＲＡＩＤシステム、テープ・ドライブ、およびデータ記録ストレージ・システムなどを含む。

【００８１】

加えて、セキュア・ソフトウェア・コンテナを作成するためのセキュア・コンテナ・システム９００を、バス・システム１００６に取り付けることができる。

【００８２】

本発明の様々な実施形態の説明は、例示の目的のために提示されているが、網羅的であることも、開示されている実施形態に限定されることも意図されていない。説明されている実施形態の範囲および思想から逸脱することなく、多くの修正および変形が当業者には明らかであろう。本明細書において使用されている用語は、実施形態の原理、実際の適用または市場に見出される技術にまさる技術的改善を最良に説明するため、または、当業者が本明細書において開示されている実施形態を理解することを可能にするために選択されている。

【００８３】

本発明は、システム、方法、またはコンピュータ・プログラム製品あるいはその組合せとして具現化することができる。コンピュータ・プログラム製品は、プロセッサに、本発明の諸態様を実行させるためのコンピュータ可読プログラム命令を有するコンピュータ可読記憶媒体を含み得る。

【００８４】

媒体は、電波媒体のための電子、磁気、光学、電磁、赤外線または半導体システムであってもよい。コンピュータ可読媒体の例は、半導体またはソリッド・ステート・メモリ、磁気テープ、着脱可能コンピュータ・ディスケット、ランダム・アクセス・メモリ（random access memory、ＲＡＭ）、読み出し専用メモリ（ＲＯＭ）、剛体磁気ディスク、および光ディスクを含み得る。光ディスクの現在の例は、コンパクト・ディスク読み出し専用メモリ（ＣＤ－ＲＯＭ）、コンパクト・ディスク・リード／ライト（ＣＤ－Ｒ／Ｗ）、ＤＶＤおよびＢｌｕ－Ｒａｙディスクを含む。

【００８５】

コンピュータ可読記憶媒体は、命令実行デバイスによって使用するための命令を保持および記憶することができる有形デバイスとすることができる。コンピュータ可読記憶媒体は例えば、限定ではないが、電子記憶デバイス、磁気記憶デバイス、光記憶デバイス、電磁記憶デバイス、半導体記憶デバイス、または上記の任意の適切な組合せであってもよい。コンピュータ可読記憶媒体のより特定的な例の包括的でないリストは、ポータブル・コンピュータ・ディスケット、ハード・ディスク、ランダム・アクセス・メモリ（ＲＡＭ）、読み出し専用メモリ（ＲＯＭ）、消去可能プログラマブル読み出し専用メモリ（ＥＰＲＯＭまたはフラッシュ・メモリ）、スタティック・ランダム・アクセス・メモリ（ＳＲＡＭ）、ポータブル・コンパクト・ディスク読み出し専用メモリ（ＣＤ－ＲＯＭ）、デジタル多用途ディスク（ＤＶＤ）、メモリ・スティック、フロッピー（Ｒ）ディスク、パンチ・カード、または、命令を記録されている溝の中の隆起構造のような機械的に符号化されているデバイス、および、上記の任意の適切な組合せを含む。コンピュータ可読記憶媒体は、本明細書において使用されるものとしては、無線波、または、他の自由に伝播する電磁波、導波路もしくは他の伝送媒体（例えば、光ファイバケーブルを通過する光パルス）を通じて伝播する電磁波、または、ワイヤを通じて伝送される電気信号のような、過渡的

信号自体として解釈されるべきではない。

【００８６】

本明細書において記載されているコンピュータ可読プログラム命令は、コンピュータ可読記憶媒体からそれぞれのコンピューティング／処理デバイスへ、または、ネットワーク、例えば、インターネット、ローカル・エリア・ネットワーク、広域ネットワークもしくはワイヤレス・ネットワークまたはその両方を介して外部コンピュータもしくは外部記憶デバイスへダウンロードすることができる。ネットワークは、銅伝送ケーブル、光伝送ファイバ、ワイヤレス送信、ルータ、ファイアウォール、スイッチ、ゲートウェイ・コンピュータまたはエッジサーバあるいはその組合せを含んでもよい。各コンピューティング／処理デバイス内のネットワーク・アダプタ・カードまたはネットワーク・インターフェースが、ネットワークからコンピュータ可読プログラム命令を受信し、それぞれのコンピューティング／処理デバイス内のコンピュータ可読記憶媒体内に記憶するために、コンピュータ可読プログラム命令を転送する。

【００８７】

本発明の動作を実行するためのコンピュータ可読プログラム命令は、アセンブラ命令、命令セットアーキテクチャ（ＩＳＡ）命令、機械命令、機械依存命令、マイクロコード、ファームウェア命令、状態設定データ、または、Ｓｍａｌｌｔａｌｋ、Ｃ＋＋などのようなオブジェクト指向プログラミング言語、および、「Ｃ」プログラミング言語もしくは同様のプログラミング言語のような従来の手続き型プログラミング言語を含む、１つまたは複数のプログラミング言語の任意の組合せで書かれているソースコードもしくはオブジェクトコードのいずれかであってもよい。コンピュータ可読プログラム命令は、その全体をユーザのコンピュータ上で、部分的にユーザのコンピュータ上で、独立型ソフトウェア・パッケージとして、部分的にユーザのコンピュータ上でかつ部分的に遠隔コンピュータ上で、またはその全体を遠隔コンピュータもしくはサーバ上で実行することができる。後者のシナリオにおいて、遠隔コンピュータが、ローカル・エリア・ネットワーク（ＬＡＮ）もしくは広域ネットワーク（ＷＡＮ）を含む任意のタイプのネットワークを通じてユーザのコンピュータに接続され、または、接続は、外部コンピュータに対して（例えば、インターネット・サービス・プロバイダを使用してインターネットを通じて）行われてもよい。いくつかの実施形態において、例えば、プログラム可能論理回路、フィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ）、またはプログラム可能論理アレイ（ＰＬＡ）を含む電子回路が、本発明の態様を実施するために、コンピュータ可読プログラム命令の状態情報を利用して電子回路をカスタマイズすることによって、コンピュータ可読プログラム命令を実行することができる。

【００８８】

本発明の態様は、本明細書において、本発明の実施形態による、方法、装置（システム）およびコンピュータ・プログラム製品のフローチャートの図またはブロック図あるいはその両方を参照して説明されている。フローチャートの図またはブロック図あるいはその両方の各ブロック、および、フローチャートの図またはブロック図あるいはその両方の中の複数のブロックの組合せはそれぞれ、コンピュータ可読プログラム製品によって実装されることができることは理解されよう。

【００８９】

これらのコンピュータ可読プログラム命令は、汎用コンピュータ、専用コンピュータ、または他のプログラム可能データ処理装置のプロセッサに提供されてマシンを生成することができ、それによって、コンピュータまたは他のプログラム可能データ処理装置のプロセッサを介して実行する命令は、フローチャートの図またはブロック図あるいはその両方の１つまたは複数のブロックにおいて指定される機能／動作を実施するための手段を作り出す。これらのコンピュータ可読プログラム命令はまた、コンピュータ、プログラム可能データ処理装置、または他のデバイスあるいはその組合せに特定の様式で機能するように指示することができるコンピュータ可読記憶媒体内に記憶することもでき、それによって、命令を記憶されているコンピュータ可読記憶媒体は、フローチャートまたはブロック図

あるいはその両方の１つまたは複数のブロックにおいて指定される機能／動作の態様を実施する命令を含む製造品を含む。

【００９０】

コンピュータ可読プログラム命令はまた、コンピュータ、他のプログラム可能データ処理装置、または他のデバイス上にロードされて、一連の動作ステップが、コンピュータ、他のプログラマブル装置、または別のデバイス上で実行されるようにして、コンピュータで実施されるプロセスを生成することができ、それによって、コンピュータ、他のプログラマブル装置、または別のデバイス上で実行する命令が、フローチャートまたはブロック図あるいはその両方の１つまたは複数のブロックにおいて指定される機能／動作を実施する。

【００９１】

図面内のフローチャートまたはブロック図あるいはその両方は本発明の様々な実施形態によるシステム、方法およびコンピュータ・プログラム製品の可能な実施態様のアーキテクチャ、機能、および動作を例示する。これに関連して、流れ図およびブロック図内の各ブロックは、指定の論理機能を実施するための１つまたは複数の実行可能命令を含む、モジュール、セグメント、または命令の一部分を表すことができる。いくつかの代替的な実施態様において、ブロックに記載されている機能は、図面に記載されている順序と一致せずに行われてもよい。例えば、連続して示されている２つのブロックは実際には、関与する機能に応じて、実質的に同時に実行されてもよく、または、これらのブロックは、時として逆順に実行されてもよい。また、ブロック図または流れ図あるいはその両方の図解の各ブロック、およびブロック図または流れ図あるいはその両方の図解のブロックの組合せは、指定の機能もしくは動作を実施するか、または、専用ハードウェアとコンピュータ命令との組合せを実行する専用ハードウェアベースのシステムによって実施することができることも留意されよう。

【００９２】

本明細書において使用される用語は特定の実施形態を説明することのみを目的とするものであり、本発明を限定するようには意図されない。本明細書において使用される場合、単数形「１つの」（“ａ”，“ａｎ”）および「その」（“ｔｈｅ”）は、別途文脈が明確に指示していない限り、複数形も含むように意図される。用語「備える」（comprises）または「備えている」（comprising）あるいはその両方は、本開示において使用されている場合、記載されている特徴、整数、ステップ、動作、要素、または構成要素あるいはその組合せが存在することを指定するが、１つまたは複数の他の特徴、整数、ステップ、動作、要素、構成要素、またはそのグループあるいはその組合せが存在することまたは追加されることを除外するものではないことがさらに理解されよう。

【００９３】

添付の特許請求の範囲内のすべてのミーンズまたはステップ・プラス・ファンクション要素の対応する構造、材料、動作、および均等物は、その機能を、他の特許請求されている要素と、具体的に特許請求されているように組み合わせて実施するための任意の構造、材料、または動作を含むように意図されている。本発明の記載は、例示および説明の目的で提示してきたものであるが、網羅的であることも、開示されている形態の本発明に限定されることも意図するものではない。本発明の範囲および思想から逸脱することなく、多くの修正および変形が当業者には明らかであろう。実施形態は本発明の原理および実際の適用を最良に説明するために、さらに当業者が、予期される特定の使用に適するように様々な改変を加えた様々な実施形態について本発明を理解することを可能にするように、選ばれ記載されている。

【００９４】

上記の要約として、本発明の一般的な概念が一連の項において説明され得る。

１．第１の項によれば、セキュア・ソフトウェア・コンテナを作成するためのコンピュータ実施方法であって、

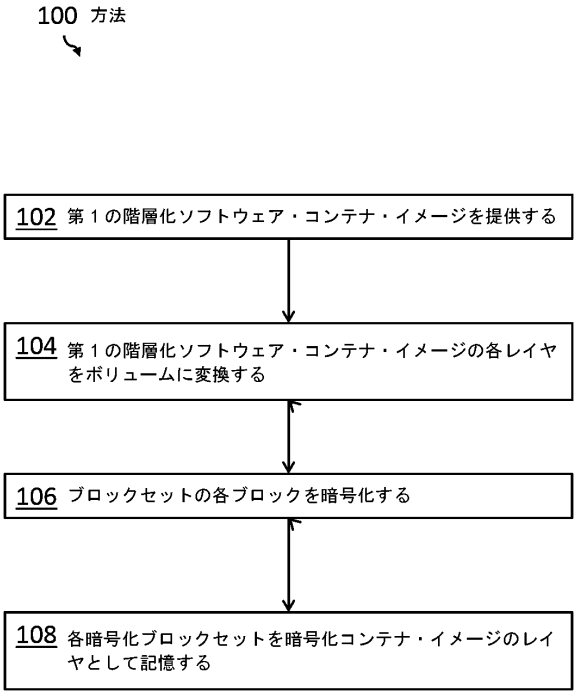
第１の階層化ソフトウェア・コンテナ・イメージを提供することと、

第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換することであり、ボリュームはブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む、変換することと、
レイヤの一部分のブロックのセットの各ブロックを暗号化することと、
ブロックの各暗号化セットを、第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックのセットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶することであり、
以て、セキュア暗号化ソフトウェア・コンテナが作成される、記憶することとを含む、コンピュータ実施方法が提供される。
２．ブロックの各暗号化セットを記憶することはまた、
第１の階層化ソフトウェア・コンテナ・イメージのメタデータを記憶することも含む、項１に記載の方法。
３．暗号化コンテナ・イメージに記憶されているレイヤの各々はシン・プロビジョニングを適用し、シン・プロビジョニング・メタデータも含む、項１または２に記載の方法。
４．暗号化コンテナ・イメージのレイヤの各ファイルの名前は、ファイルの内容のハッシュ値である、項１ないし３のいずれかに記載の方法。
５．仮想機械オペレーティング・システム、開始プログラム、および復号キーを提供することも含み、復号キーは、ブロックのセットの各ブロックの暗号化に使用されている暗号化キーに対応する、項１ないし４のいずれかに記載の方法。
６．仮想機械オペレーティング・システムによる仮想機械の開始に有効であるセキュア・コンテナ実行環境を提供することも含む、項５に記載の方法。
７．仮想機械の開始は、
復号キーを使用して暗号化コンテナ・イメージのブロックのセットを復号することを含む、項６に記載の方法。
８．第１の階層化ソフトウェア・コンテナ・イメージのレイヤのシーケンスにおいて第１の階層化ソフトウェア・コンテナ・イメージを再構築することも含む、項７に記載の方法。
９．復号されているセキュア・コンテナ・イメージの上のレイヤが、リード／ライト・アクセスを可能にし、当該最上レイヤの下のレイヤが、リード・オンリー・アクセスを可能にする、項８に記載の方法。
１０．セキュア・コンテナ実行環境は、権限のあるユーザまたは他のプロセスあるいはその両方による仮想機械へのアクセスを妨げるセキュア・ファームウェアによって保護され、仮想機械オペレーティング・システム、開始プログラムおよび復号キーが各々暗号化される、項５ないし９に記載の方法。
１１．セキュア・ファームウェアは、ハードウェア・セキュリティ・モジュールと協働する、項１０に記載の方法。
１２．セキュア・ソフトウェア・コンテナを作成するためのセキュア・コンテナ・システムであって、
第１の階層化ソフトウェア・コンテナ・イメージを受信するように適合されている受信ユニットと、
第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換するように適合されている変換ユニットであり、ボリュームはブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む、変換ユニットと、
レイヤの一部分のブロックのセットの各ブロックを暗号化するように適合されている暗号化モジュールと、
ブロックの各暗号化セットを、第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックのセットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶するように適合されている記憶ユニットであり、
以て、セキュア暗号化ソフトウェア・コンテナが作成される、記憶ユニットとを備える、セキュア・コンテナ・システムも提供される。

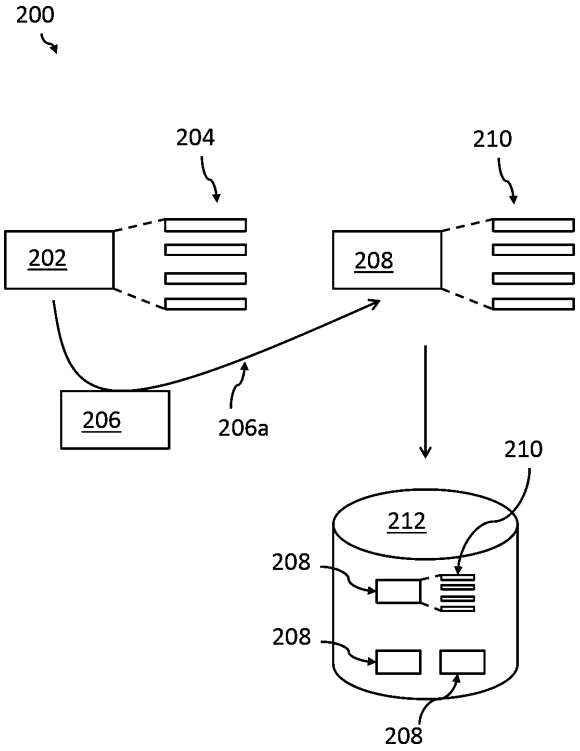
１３．記憶ユニットはまた、
　第１の階層化ソフトウェア・コンテナ・イメージのメタデータを記憶するようにも適合されている、項１２に記載のシステム。
１４．暗号化コンテナ・イメージに記憶されているレイヤの各々はシン・プロビジョニングを使用し、シン・プロビジョニング・メタデータも含む、項１２または１３に記載のシステム。
１５．暗号化コンテナ・イメージのレイヤの各ファイルの名前は、ファイルの内容のハッシュ値である、項１２ないし１４のいずれかに記載のシステム。
１６．仮想機械オペレーティング・システム、開始プログラム、および復号キーを提供するように適合されている提供モジュールも備え、復号キーは、ブロックのセットの各ブロックの暗号化に使用されている暗号化キーに対応する、項１２ないし１５のいずれかに記載のシステム。
１７．提供モジュールＩはまた、
　仮想機械オペレーティング・システムによる仮想機械の開始に有効であるセキュア・コンテナ実行環境を提供するようにも適合されている、項１６に記載のシステム。
１８．仮想機械の開始は、
　復号キーを使用して暗号化コンテナ・イメージのブロックのセットを復号することを含む、項１７に記載のシステム。
１９．第１の階層化ソフトウェア・コンテナ・イメージのレイヤのシーケンスにおいて第１の階層化ソフトウェア・コンテナ・イメージを再構築するように適合されている再構築ユニットも備える、項１８に記載のシステム。
２０．復号されているセキュア・コンテナ・イメージの上のレイヤが、リード／ライト・アクセスを可能にし、当該最上レイヤの下のレイヤが、リード・オンリー・アクセスを可能にする、項１９に記載のシステム。
２１．セキュア・コンテナ実行環境は、権限のあるユーザまたは他のプロセスあるいはその両方による仮想機械へのアクセスを妨げるセキュア・ファームウェアによって保護され、仮想機械オペレーティング・システム、開始プログラムおよび復号キーが各々暗号化される、項１６ないし２０のいずれかに記載のシステム。
２２．セキュア・ファームウェアと協働するように適合されているハードウェア・セキュリティ・モジュールも備える、項２１に記載のシステム。
２３．セキュア・ソフトウェア・コンテナを作成するためのコンピュータ・プログラム製品であって、プログラム命令を具現化されているコンピュータ可読記憶媒体を備え、プログラム命令は、１つまたは複数のコンピューティング・システムまたはコントローラによって、１つまたは複数のコンピューティング・システムに、
　第１の階層化ソフトウェア・コンテナ・イメージを提供することと、
　第１の階層化ソフトウェア・コンテナ・イメージの各レイヤの、対応するメタデータを除くすべてのファイルをボリュームに変換することであり、ボリュームはブロックのセットを含み、各レイヤは、次により低いレイヤに対する漸進的な差を含む、変換することと、
　レイヤの一部分のブロックのセットの各ブロックを暗号化することと、
　ブロックの各暗号化セットを、第１の階層化ソフトウェア・コンテナ・イメージの順序に等しいブロックのセットの順序を再構築するための非暗号化メタデータとともに、暗号化コンテナ・イメージのレイヤとして記憶することであり、
　以て、セキュア暗号化ソフトウェア・コンテナが作成される、記憶することとを行わせるように実行可能である、コンピュータ・プログラム製品。

【図面】

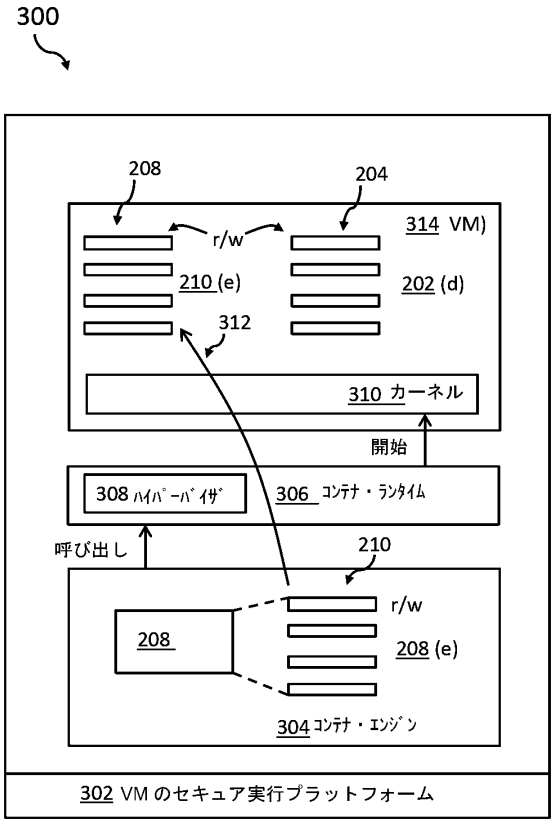
【図１】


100 方法
102 第１の階層化ソフトウェア・コンテナ・イメージを提供する
104 第１の階層化ソフトウェア・コンテナ・イメージの各レイヤをボリュームに変換する
106 ブロックセットの各ブロックを暗号化する
108 各暗号化ブロックセットを暗号化コンテナ・イメージのレイヤとして記憶する

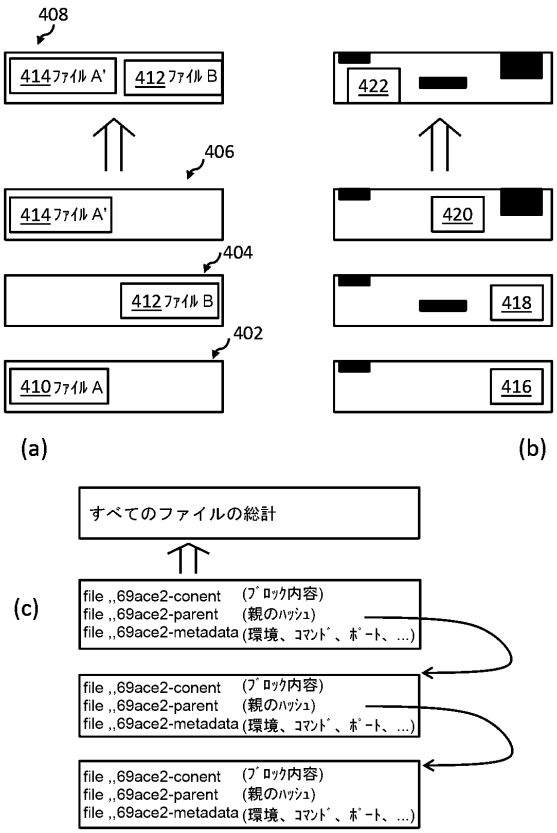
【図２】


200
204
210
202
208
206
206a
212
208
208
208

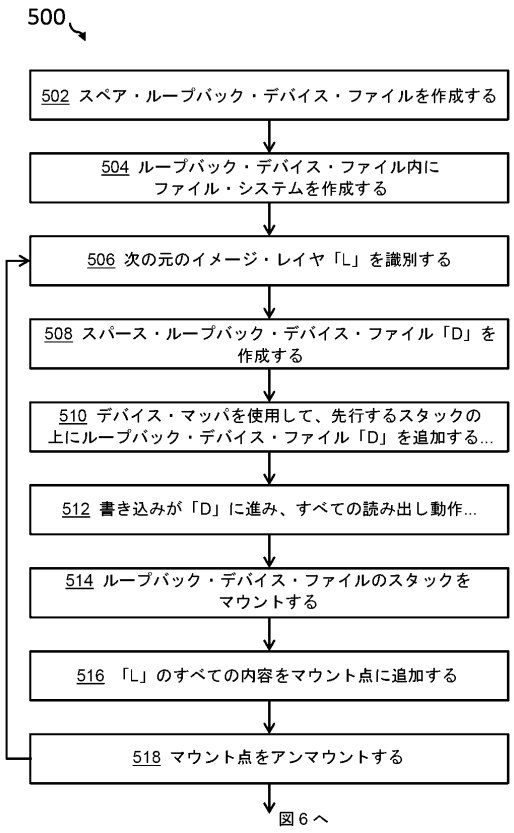
【図３】


300
208
204
314 VM)
r/w
210 (e)
202 (d)
312
310 カーネル
開始
308 ハイパーバイザ
306 コンテナ・ランタイム
呼び出し
210
r/w
208
208 (e)
304 コンテナ・エンジン
302 VM のセキュア実行プラットフォーム

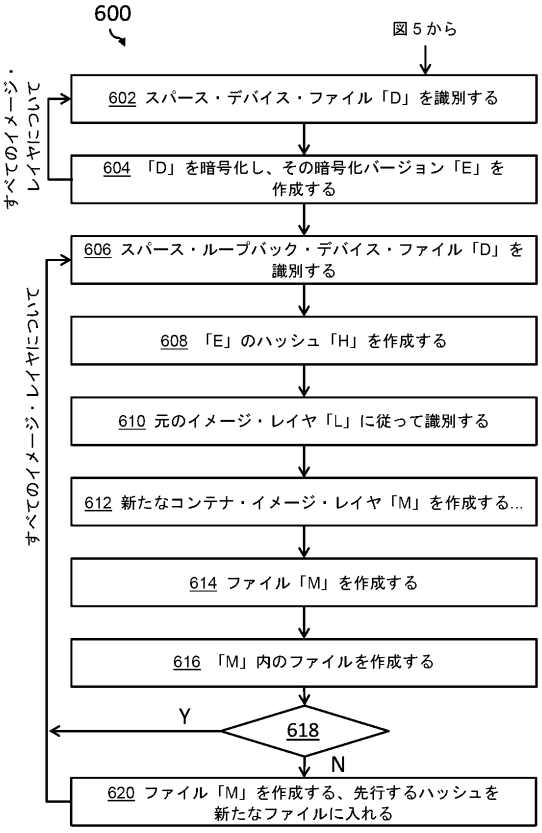
【図４】


408
414 ファイル A'
412 ファイル B
406
414 ファイル A'
404
412 ファイル B
402
410 ファイル A
(a)
422
420
418
416
(b)
すべてのファイルの総計
(c)
file ,,69ace2-conent (ブロック内容)
file ,,69ace2-parent (親のハッシュ)
file ,,69ace2-metadata (環境、コマンド、ポート、...)
file ,,69ace2-conent (ブロック内容)
file ,,69ace2-parent (親のハッシュ)
file ,,69ace2-metadata (環境、コマンド、ポート、...)
file ,,69ace2-conent (ブロック内容)
file ,,69ace2-parent (親のハッシュ)
file ,,69ace2-metadata (環境、コマンド、ポート、...)

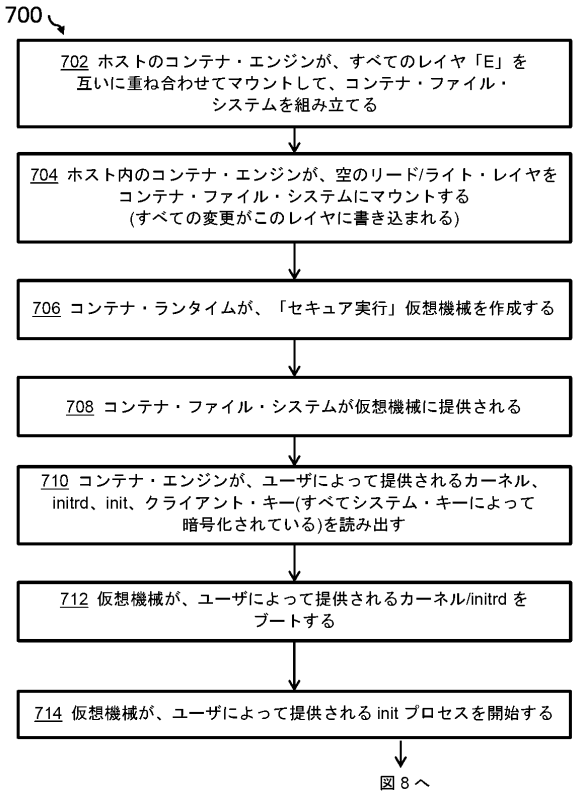
【図５】


500
502 スペア・ループバック・デバイス・ファイルを作成する
504 ループバック・デバイス・ファイル内にファイル・システムを作成する
506 次の元のイメージ・レイヤ「L」を識別する
508 スパース・ループバック・デバイス・ファイル「D」を作成する
510 デバイス・マッパを使用して、先行するスタックの上にループバック・デバイス・ファイル「D」を追加する...
512 書き込みが「D」に進み、すべての読み出し動作...
514 ループバック・デバイス・ファイルのスタックをマウントする
516 「L」のすべての内容をマウント点に追加する
518 マウント点をアンマウントする
図６へ

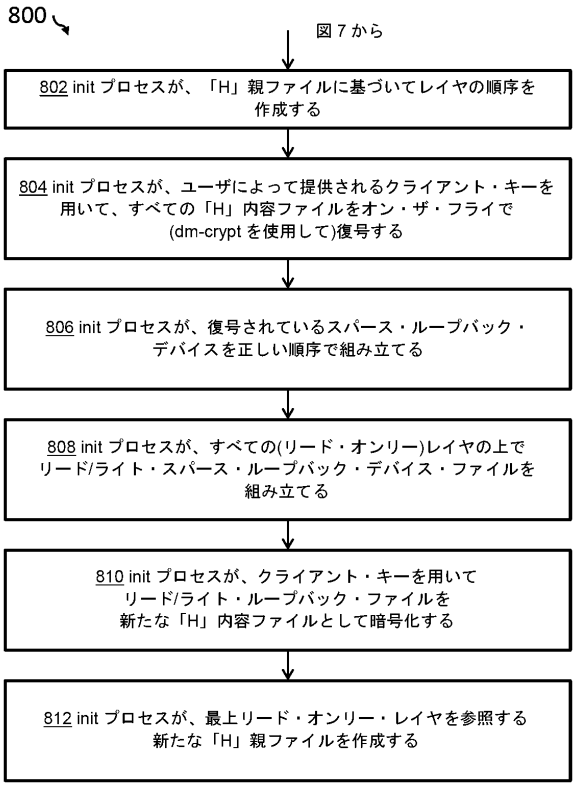
【図６】


600
図５から
すべてのイメージ・レイヤについて
602 スパース・デバイス・ファイル「D」を識別する
604 「D」を暗号化し、その暗号化バージョン「E」を作成する
すべてのイメージ・レイヤについて
606 スパース・ループバック・デバイス・ファイル「D」を識別する
608 「E」のハッシュ「H」を作成する
610 元のイメージ・レイヤ「L」に従って識別する
612 新たなコンテナ・イメージ・レイヤ「M」を作成する...
614 ファイル「M」を作成する
616 「M」内のファイルを作成する
618
Y
N
620 ファイル「M」を作成する、先行するハッシュを新たなファイルに入れる

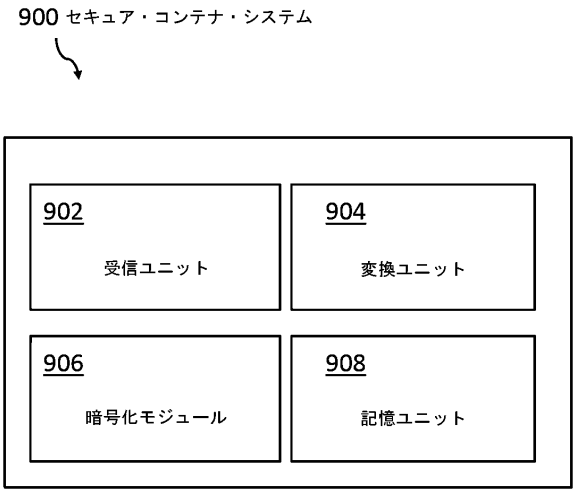
【図７】


700
702 ホストのコンテナ・エンジンが、すべてのレイヤ「E」を互いに重ね合わせてマウントして、コンテナ・ファイル・システムを組み立てる
704 ホスト内のコンテナ・エンジンが、空のリード/ライト・レイヤをコンテナ・ファイル・システムにマウントする
(すべての変更がこのレイヤに書き込まれる)
706 コンテナ・ランタイムが、「セキュア実行」仮想機械を作成する
708 コンテナ・ファイル・システムが仮想機械に提供される
710 コンテナ・エンジンが、ユーザによって提供されるカーネル、initrd、init、クライアント・キー(すべてシステム・キーによって暗号化されている)を読み出す
712 仮想機械が、ユーザによって提供されるカーネル/initrd をブートする
714 仮想機械が、ユーザによって提供される init プロセスを開始する
図８へ

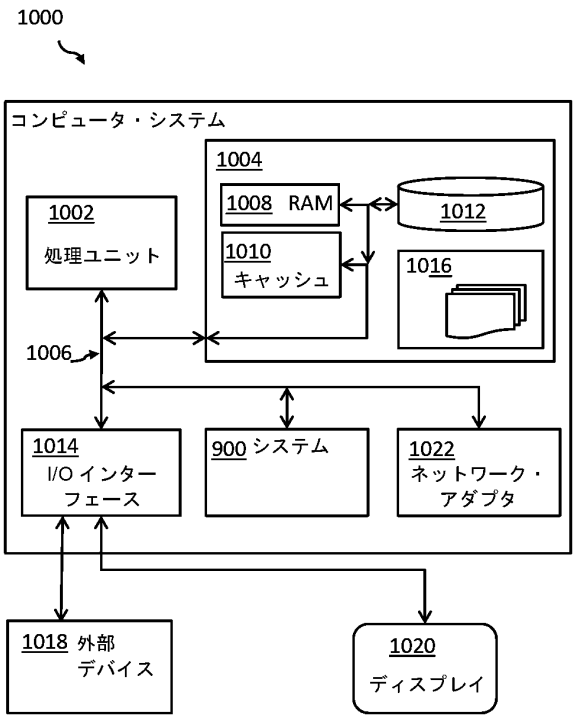
【図８】


800
図７から
802 init プロセスが、「H」親ファイルに基づいてレイヤの順序を作成する
804 init プロセスが、ユーザによって提供されるクライアント・キーを用いて、すべての「H」内容ファイルをオン・ザ・フライで(dm-crypt を使用して)復号する
806 init プロセスが、復号されているスパース・ループバック・デバイスを正しい順序で組み立てる
808 init プロセスが、すべての(リード・オンリー)レイヤの上でリード/ライト・スパース・ループバック・デバイス・ファイルを組み立てる
810 init プロセスが、クライアント・キーを用いてリード/ライト・ループバック・ファイルを新たな「H」内容ファイルとして暗号化する
812 init プロセスが、最上リード・オンリー・レイヤを参照する新たな「H」親ファイルを作成する

【図９】


900 セキュア・コンテナ・システム
902
受信ユニット
904
変換ユニット
906
暗号化モジュール
908
記憶ユニット

【図１０】


1000
コンピュータ・システム
1004
1008 RAM
1012
1002
処理ユニット
1010
キャッシュ
1016
1006
1014
I/O インター
フェース
900 システム
1022
ネットワーク・
アダプタ
1018 外部
デバイス
1020
ディスプレイ

フロントページの続き

(72)発明者　バッヒャー、ウッツ
ドイツ　７１０３２　ベーブリンゲン　シェーナイヒャー・シュトラーセ　２２０
(72)発明者　ブエントゲン、ラインハルト
ドイツ　７１０３２　ベーブリンゲン　シェーナイヒャー・シュトラーセ　２２０
(72)発明者　モルジャン、ピーター
ドイツ　７１０３２　ベーブリンゲン　シェーナイヒャー・シュトラーセ　２２０
(72)発明者　フランク、ヤーノシュ
ドイツ　７１０３２　ベーブリンゲン　シェーナイヒャー・シュトラーセ　２２０

審査官　児玉 崇晶

(56)参考文献　特開２０１７－１３０１９２（ＪＰ，Ａ）
米国特許出願公開第２０１７／０１７７８６０（ＵＳ，Ａ１）
西島 剛，はじめてのＤｏｃｋｅｒ，初版，株式会社工学社，2016年04月25日，pp.7-10, 20-23, 25-26, 34-37, 68-70
佐藤 寛文，ＯｖｅｒｌａｙＦＳを用いたコンテナに対するサービス妨害攻撃の防止，コンピュータセキュリティシンポジウム２０１８論文集，一般社団法人情報処理学会，2018年10月25日，pp.38-45

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／４４５
Ｇ０６Ｆ　９／４５５
Ｇ０６Ｆ　２１／６０
Ｇ０６Ｆ　２１／６２
Ｇ０６Ｆ　２１／５３