(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2020-505701
(P2020-505701A)
(43)公表日　令和2年2月20日(2020.2.20)

| (51)Int.Cl. | F I | テーマコード（参考） |
|---|---|---|
| G06F 21/57 (2013.01) | G06F 21/57 | |

審査請求 未請求　予備審査請求 未請求　（全 64 頁）

(21)出願番号　特願2019-540009 (P2019-540009)
(86)(22)出願日　平成29年12月20日 (2017.12.20)
(85)翻訳文提出日　令和1年8月20日 (2019.8.20)
(86)国際出願番号　PCT/US2017/067451
(87)国際公開番号　WO2018/140160
(87)国際公開日　平成30年8月2日 (2018.8.2)
(31)優先権主張番号　15/414,355
(32)優先日　平成29年1月24日 (2017.1.24)
(33)優先権主張国・地域又は機関
米国 (US)

(71)出願人　314015767
マイクロソフト　テクノロジー　ライセンシング，エルエルシー
アメリカ合衆国　ワシントン州　９８０５２　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100118902
弁理士　山本　修
(74)代理人　100106208
弁理士　宮前　徹
(74)代理人　100120112
弁理士　中西　基晴

最終頁に続く

(54)【発明の名称】抽象エンクレーブアイデンティティ

(57)【要約】
抽象エンクレーブアイデンティティが提示される。抽象アイデンティティは、複数の関連するが同一ではないエンクレーブインスタンス化では同じであり得るセキュアなアイデンティティであり得る。エンクレーブアイデンティティ値は、インスタンス化されたエンクレーブに関する抽象エンクレーブアイデンティティタイプから決定され得る。様々なエンクレーブ動作は、データを抽象アイデンティティへ密封すること、単調カウンターを増分すること、信頼できる時刻測定を行うことなど、抽象アイデンティティを用いて実施され得る。

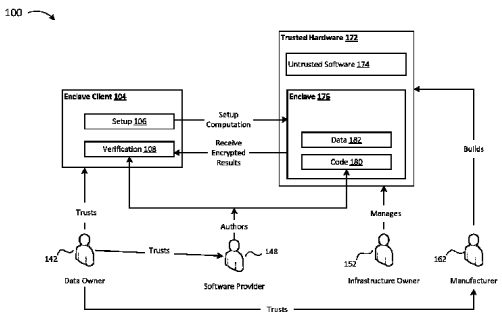

FIG. 1

【特許請求の範囲】
【請求項１】
プロセッサーおよびメモリを備えるコンピューティングデバイスによって実施されるエンクレーブ(enclave)識別方法であって、
アイデンティティタイプ、およびインスタンス化されたエンクレーブに関連した動作のリクエストを受信するステップと、
前記アイデンティティタイプ、および前記エンクレーブがインスタンス化されたエンクレーブ画像から導出される情報に基づいて、前記エンクレーブのアイデンティティ値を決定するステップと、
前記アイデンティティ値を用いて前記動作を実施するステップと、を含む方法。
【請求項２】
前記エンクレーブ画像のオーサー(author)に関連した公開鍵を用いて前記エンクレーブ画像内の署名を検証することによって、前記アイデンティティ値の完全性を検証するステップをさらに含む、請求項１に記載の方法。
【請求項３】
前記アイデンティティ値が、ハッシュ関数の出力として決定され、また
前記アイデンティティタイプに少なくとも部分的に基づいて、前記エンクレーブ画像のアイデンティティ部を決定するステップと、
前記アイデンティティ部について前記ハッシュ関数をコンピューティングするステップとをさらに含む、請求項１に記載の方法。
【請求項４】
前記エンクレーブ画像が、追加の依存エンクレーブ画像への参照を含み、
前記アイデンティティタイプに部分的に基づいて、追加の依存エンクレーブ画像のうちのどれが前記ハッシュ関数への入力として含まれるかを決定するステップをさらに含む、請求項３に記載の方法。
【請求項５】
前記アイデンティティ部が、前記インスタンス化されたエンクレーブのインスタンス化中にセキュアエンクレーブコンテナ内へコピーされるバイナリコード、および実行可能コードではない１つまたは複数の識別子、のうちの１つまたは複数を含む、請求項３に記載の方法。
【請求項６】
前記アイデンティティ値が、ハッシュ関数の出力として決定され、
前記エンクレーブが、複数のエンクレーブ画像を用いてインスタンス化されており、
前記アイデンティティタイプが、前記複数のエンクレーブ画像のうちのどれが、前記アイデンティティ値を決定するために前記ハッシュ関数に少なくとも部分的に含まれるかを示す、請求項１に記載の方法。
【請求項７】
前記動作が、前記アイデンティティ値を有するアテステーションレポートを生成することを含む、請求項１に記載の方法。
【請求項８】
前記動作が、前記アイデンティティ値を有するデータを密封する(sealing)ことによって、データを前記エンクレーブへ密封することを含む、請求項１に記載の方法。
【請求項９】
前記動作が、単調カウンターを増分することを含み、前記単調カウンターが、前記アイデンティティ値によって識別される、請求項１に記載の方法。
【請求項１０】
前記動作が、信頼できる時刻測定を行うことを含み、前記信頼できる時刻が、前記アイデンティティ値に基づいて決定される、請求項１に記載の方法。
【請求項１１】
少なくともプロセッサーと命令を格納するメモリとを備えるシステムであって、前記命

令が、前記システムによって実行されるとき、少なくとも
アイデンティティタイプ、およびインスタンス化されたエンクレーブに関連した動作のリクエストを受信することと、
前記アイデンティティタイプ、および前記エンクレーブがインスタンス化されたエンクレーブ画像から導出される情報に基づいて、前記エンクレーブのアイデンティティ値を決定することと、
前記アイデンティティ値を用いて前記動作を実施することとを引き起こす、システム。
【請求項１２】
前記命令が、少なくとも
前記エンクレーブ画像のオーサーに関連した公開鍵を用いて前記エンクレーブ画像内の署名を検証することによって、前記アイデンティティ値の完全性を検証することをさらに引き起こす、請求項１１に記載のシステム。
【請求項１３】
前記アイデンティティ値が、ハッシュ関数の出力として決定され、また前記命令が、少なくとも
前記アイデンティティタイプに少なくとも部分的に基づいて、前記エンクレーブ画像のアイデンティティ部を決定することと、
前記アイデンティティ部について前記ハッシュ関数をコンピューティングすることとをさらに引き起こす、請求項１１に記載のシステム。
【請求項１４】
前記エンクレーブ画像が、追加の依存エンクレーブ画像への参照を含み、また前記命令が、少なくとも
前記アイデンティティタイプに部分的に基づいて、追加の依存エンクレーブ画像のうちのどれが前記ハッシュ関数への入力として含まれるかを決定することをさらに引き起こす、請求項１３に記載のシステム。
【請求項１５】
アイデンティティタイプ、およびインスタンス化されたエンクレーブに関連した動作のリクエストを受信することと、
前記アイデンティティタイプ、および前記エンクレーブのコンテナに格納されるデータに基づいて、前記エンクレーブのアイデンティティ値を決定することと、
前記アイデンティティ値を用いて前記動作を実施することと、を行うように構成される、コンピューティングデバイス。
【発明の詳細な説明】
【技術分野】
【０００１】
[0001]本開示は、コンピューティングシステムを守ることに関する。
【背景技術】
【０００２】
[0002]セキュアな隔離領域または信頼できる実行環境は、隔離領域の外側の領域内にあまり信頼できないコードも有し得るコンピューター上で信頼できるコードを実行するための、本明細書ではエンクレーブと称されるセキュアコンテナを提供する。エンクレーブの隔離領域は、エンクレーブの外側に存在するコードの実行中に保護されるメモリの一部分を含む。隔離されたメモリは、エンクレーブのためのコードおよびデータの両方を含み得、このメモリの保護は、エンクレーブメモリからの読み込みまたはそこへの書き込みに対する制限に加えて、エンクレーブメモリ内に含まれるコードを実行する制限を含み得る。メモリ隔離および実行制限などのエンクレーブセキュリティの面は、例えば、コンピュータープロセッサー内のハードウェアによって実施され得る。ソフトウェアアテステーションは、特定のエンクレーブの隔離セキュリティ、およびその特定のエンクレーブの隔離されたメモリ領域内にロードされるエンクレーブコードにおける信頼を提供し得る。アテステーションは、追加で、証明されたエンクレーブが実行しているハードウェアおよびソフ

トウェアプラットフォームの完全性の証拠を提供し得る。
【０００３】
[0003]ＭｉｃｒｏｓｏｆｔのＶｉｒｔｕａｌ Ｓｅｃｕｒｅ Ｍｏｄｅ（ＶＳＭ）およびＩｎｔｅｌのＳｏｆｔｗａｒｅ Ｇｕａｒｄ Ｅｘｔｅｎｓｉｏｎｓ（ＳＧＸ）などのエンクレーブシステムは、部分的には、ユーザーモードまたはカーネルモードのいずれかで実行する他のコードからエンクレーブを隔離することによって、セキュリティを提供する。完全性および機密性保証は、エンクレーブ内で実行するコードの真正性における高レベルの信頼、およびエンクレーブコードの安全な実行における信頼をエンクレーブに提供し得る。完全性保証は、特定のエンクレーブのソフトウェアアテステーションによって提供され得る。ソフトウェアアテステーションは、エンクレーブの内側のコンテンツ（命令およびデータ）の暗号化署名付きハッシュを含み得、エンクレーブ環境に関するデータと組み合わされ得る。エンクレーブが、Ｔｒｕｓｔｅｄ Ｃｏｍｐｕｔｉｎｇ Ｇｒｏｕｐ（ＴＣＧ）トラステッドプラットフォームモジュール（ＴＰＭ）規格に準拠するハードウェアなど、ハードウェアセキュリティモジュール（ＨＳＭ）と組み合わせて使用されるとき、エンクレーブは、付加的レベルのセキュリティおよび機密性保証を提供することができる。
【０００４】
[0004]エンクレーブの隔離の外側の信頼できないローカルコードからの信頼できるローカルエンクレーブの隔離により提供されるセキュリティに加えて、エンクレーブのソフトウェアアテステーションは、リモートの信頼できるコンピューティングを可能にすることができる。リモートエンクレーブのアテステーションは、エンクレーブ内での命令の実行の完全性、ならびにエンクレーブによって処理されるデータ機密性の両方において信頼を提供し得る。リモートエンクレーブのアテステーションが信頼できる製造業者からのハードウェアによって提供されるとき、エンクレーブは、非トラステッドパーティーによって所有および管理される未知のコンピューター上にエンクレーブが存在するときにさえ、信頼され得る。これは多くの場合、例えば、コンピューティングリソースが、インターネットクラウドベースのコンピューティングリソース上でレンタルされるときに当てはまる。
【発明の概要】
【発明が解決しようとする課題】
【０００５】
本願発明の一実施例は、例えば、抽象エンクレーブアイデンティティに関する。
【課題を解決するための手段】
【０００６】
[0005]エンクレーブプラットフォームを抽象化するための方法およびシステムが提示される。本方法は、エンクレーブ抽象化プラットフォームによって、エンクレーブクライアントからエンクレーブを使用するための第１のリクエストを受信することを含み得る。第１のリクエストは、クライアント抽象化プロトコルに準拠し得る。第１のリクエストは、エンクレーブネイティブプラットフォームと関連付けられたエンクレーブネイティブプロトコルに準拠する第２のリクエストに変換され得る。次いで、第２のリクエストは、エンクレーブネイティブプラットフォームに送信され得る。第１のリクエストは、例えば、エンクレーブをインスタンス化するためのリクエスト、エンクレーブのアテステーションレポートを検証するためのリクエスト、エンクレーブへコールインするためのリクエスト、またはエンクレーブおよびエンクレーブクライアントの両方と共有されるメモリを割り当てるためのリクエストであり得る。ネイティブプラットフォームは、Ｉｎｔｅｌ Ｓｏｆｔｗａｒｅ Ｇｕａｒｄ Ｅｘｔｅｎｓｉｏｎｓ（ＳＧＸ）エンクレーブアーキテクチャに準拠し得、ネイティブプラットフォームは、Ｍｉｃｒｏｓｏｆｔ Ｖｉｒｔｕａｌ Ｓｅｃｕｒｅ Ｍｏｄｅ（ＶＳＭ）エンクレーブアーキテクチャに準拠し得る。
【図面の簡単な説明】
【０００７】
【図１】[0006]エンクレーブシステムの例示的なハイレベルブロック図である。

【図２】[0007]機密性保証付きのメッセージ引き渡しのための例示的なプロセスを描写する図である。
【図３】[0008]完全性保証付きのメッセージ引き渡しのための例示的なプロセスを描写する図である。
【図４】[0009]新鮮性保証付きのメッセージ引き渡しのための例示的なプロセスを描写する図である。
【図５】[0010]エンクレーブのソフトウェアアテステーションのための例示的なプロセスを描写する図である。
【図６】[0011]例示的なディフィー・ヘルマン鍵交換（ＤＫＥ）プロトコルを描写する図である。
【図７】[0012]ソフトウェアアテステーションのための例示的な信頼チェーンを描写する図である。
【図８】[0013]例示的なローカルエンクレーブシステムのためのソフトウェアコンポーネントインターフェースのブロック図である。
【図９】[0014]抽象化層を有する例示的なローカルエンクレーブシステムのためのソフトウェアコンポーネントインターフェースのブロック図である。
【図１０】[0015]抽象化層を有する例示的なリモートエンクレーブシステムのためのソフトウェアコンポーネントインターフェースのブロック図である。
【図１１】[0016]例示的な汎用コンピューティング環境を描写する図である。
【図１２】[0017]ネイティブエンクレーブプラットフォームを抽象化する方法の例示的なフローチャートを描写する図である。
【図１３】[0018]ネイティブエンクレーブプラットフォームを抽象化する方法の例示的なフローチャートを描写する図である。
【図１４】[0019]抽象エンクレーブアイデンティティを用いてエンクレーブ動作を実施する方法の例示的なフローチャートを描写する図である。
【図１５】[0020]抽象エンクレーブアイデンティティを用いてエンクレーブ動作を実施する方法１５００の例示的なフローチャートを描写する図である。
【図１６】[0021]抽象エンクレーブアイデンティティ等価性を有する例示的なシステムを描写する図である。
【図１７】[0022]２つの等価のエンクレーブを用いた並列処理の例示的なフローチャートを描写する図である。
【図１８】[0023]２つの等価のエンクレーブを用いた逐次処理の例示的なフローチャートを描写する図である。
【図１９】[0024]例示的な分散データ密封システムのブロック図である。
【図２０】[0025]分散データ密封および開封の例示的なフローチャートである。
【図２１】[0026]例示的なＫｅｙ　Ｖａｕｌｔエンクレーブのブロック図である。
【図２２】[0027]いくつかのＫｅｙ　Ｖａｕｌｔエンクレーブ動作の例示的なフローチャートである。
【図２３】[0028]Ｖａｕｌｔロックされた鍵を用いたＫｅｙ　Ｖａｕｌｔエンクレーブ動作の例示的なフローチャートである。
【発明を実施するための形態】
【０００８】
[0029]エンクレーブクライアントの開発、およびエンクレーブ内で実行するソフトウェアの開発を簡略化するエンクレーブの抽象化モデルが開示される。抽象化モデルは、ＩｎｔｅｌのＳＧＸおよびＭｉｃｒｏｓｏｆｔのＶＳＭなどのネイティブエンクレーブプラットフォーム・アーキテクチャの簡略化および一体化であり得る。抽象化層ソフトウェアコンポーネントは、エンクレーブクライアントと１つまたは複数のネイティブプラットフォームとの間、エンクレーブの内側のソフトウェアと１つまたは複数のネイティブエンクレーブプラットフォームとの間、およびエンクレーブの内側のソフトウェアとエンクレーブクライアントとの間の通信を翻訳し得る。そのような抽象化プラットフォームは、エンク

レーブソフトウェアおよびエンクレーブクライアントソフトウェアのシングルバージョンがＳＧＸおよびＶＳＭなどの複数のネイティブエンクレーブプラットフォームの上で実行することを可能にするという利益を提供し得る。エンクレーブおよびエンクレーブクライアントのための書き込みソフトウェアのタスクを簡略化することに加えて、これは、エンクレーブのエンドユーザーが、特定のコンピューターのネイティブエンクレーブプラットフォームのために作られるエンクレーブおよびエンクレーブクライアントソフトウェアの両方のバージョンを探し出す必要性なしに、任意のサポート対象のネイティブエンクレーブアーキテクチャをサポートするコンピューター上でエンクレーブおよびエンクレーブクライアントソフトウェアを実行することを可能にする。

【０００９】

[0030]エンクレーブ抽象化モデルは、例えば、エンクレーブのライフサイクルを管理すること、エンクレーブのローカルおよびリモートアテステーション、エンクレーブへデータを密封すること、エンクレーブ内へのおよびエンクレーブの外へのプログラム転送制御、ならびに単調カウンターおよび信頼できる時刻などの他のセキュリティ特徴のためのプリミティブを含む。エンクレーブのコンテンツのシングルバイナリまたはシングルハッシュの域を越えたエンクレーブのアイデンティティを抽象化するエンクレーブの抽象または層状アイデンティティも提示される。アプリケーションプログラミングインターフェース（ＡＰＩ）またはアプリケーションバイナリインターフェース（ＡＢＩ）などのソフトウェアコンポーネントインターフェースは、抽象化モデルプリミティブを使用したエンクレーブおよびエンクレーブクライアントプログラムの開発のために提示される。

【００１０】

[0031]抽象アイデンティティは、エンクレーブインスタンスのグループをセキュアに識別するために使用され得るネスト化アイデンティティまたはアイデンティティ階層を含み得る。本明細書におけるエンクレーブインスタンスは、同じマシン上のエンクレーブ内へロードされる同じエンクレーブバイナリコードを指し得、同じアイデンティティを有し得る。その一方で、バイナリコードの新しいバージョン、または異なるマシン上にロードされる同じバイナリコードは、異なるインスタンスと見なされ得る。これらの異なるインスタンスはまた、アイデンティティ階層内のより高いレベルにおいては同じアイデンティティを有し得る。抽象化されたエンクレーブアイデンティティは、関連したエンクレーブバイナリのグループが関連するものとして識別されることを可能にする。例えば、バグが修正される前後のエンクレーブバイナリのバージョンなど、同じエンクレーブの異なるバージョンは、バージョンとは無関係に、同じ名前を与えられ得る。抽象化のより高い層において、エンクレーブのファミリー内のすべてのエンクレーブは、単一のファミリー名またはアイデンティティを与えられ得る。この場合、関連するが異なる関数を実施するすべてのエンクレーブは、一緒に識別され得る。他のアイデンティティ層またはアイデンティティのグループ分けは、以下に説明される。

【００１１】

[0032]抽象アイデンティティの任意の層は、データを密封すること、エンクレーブのアテステーション、またはデータ新鮮性を保証すること（単調カウンターによる）など、様々な暗号化動作のために使用され得る。例えば、１つのエンクレーブインスタンスによってもたらされるデータをより高いレベルのアイデンティティへ密封することによって、その後そのデータは、同じより高レベルのエンクレーブアイデンティティを有する異なるエンクレーブインスタンスによってセキュアに消費され得る。例えば、エンクレーブファミリーへデータを密封することによって、そのファミリーのメンバーである任意のエンクレーブインスタンスが、およびそのファミリーのメンバーのみが、そのデータを開封することができることになる。エンクレーブインスタンスからのファミリーアイデンティティへのアテステーションは、そのエンクレーブインスタンスがファミリーのメンバーであるということを保証する。アイデンティティ抽象化に結び付けられる単調カウンターは、抽象化アイデンティティのメンバーであるすべてのエンクレーブインスタンスに関連する新鮮性保証を提供し得る。

【００１２】
[0033]開示される抽象化モデルは、アプリケーションプログラミングインターフェース（ＡＰＩ）またはアプリケーションバイナリインターフェース（ＡＢＩ）など、エンクレーブおよびエンクレーブホストのソフトウェア開発を簡略化し得るソフトウェアコンポーネントインターフェースを含む。ＡＰＩは、プログラミングサブルーチン定義、プロトコル、およびソフトウェを作成するためのツールのセットである。ＡＰＩは、ソフトウェアコンポーネントの特定の実装形態とは無関係に、ソフトウェアコンポーネントの入力および出力、ソフトウェアコンポーネントによって使用されるデータタイプ、ならびにソフトウェアコンポーネントの機能性または動作を規定し得る。ＡＰＩは、Ｃ、Ｃ＋＋、Ｃ＃、および同様のものなど、高水準コンピューター言語で規定され得る。ＡＰＩの形式化された定義は、ソフトウェアコンポーネント間、例えば、異なる時刻に、または異なるオーサーによって書かれた２つのソフトウェアコンポーネント間の相互作用を促進し得る。ＡＰＩは、Ｍｉｃｒｏｓｏｆｔインターフェース定義原語（ＭＩＤＬ）またはオブジェクトマネージメントグループ（ＯＭＧ）ＩＤＬなどのインターフェース記述言語（ＩＤＬ）を部分的に用いて形式化され得る。ＡＢＩはまた、ソフトウェアコンポーネント間のインターフェースであるが、オブジェクトコードインターフェースである。例えば、ＡＢＩは、オブジェクトコードのエントリーポイント（または命令アドレス）であり得、このオブジェクトコードは、エントリーポイントが呼び出されるときに、引数を保持するマシンレジスタを特定するプロトコルなど、それらのエントリーポイントを使用するためのプロトコルと一緒にＡＰＩのソースコード実装をコンパイルすることから生じる。

【００１３】
[0034]上に説明されるように異なるレベルのエンクレーブアイデンティティとの相互作用を可能にすることに加えて、エンクレーブＡＰＩは、エンクレーブプラットフォームアーキテクチャ間、例えば、ＩｎｔｅｌのＳｏｆｔｗａｒｅ Ｇｕａｒｄ Ｅｘｔｅｎｓｉｏｎｓ（ＳＧＸ）、ＭｉｃｒｏｓｏｆｔのＶｉｒｔｕａｌ Ｓｅｃｕｒｅ Ｍｏｄｅ（ＶＳＭ）、およびＡＲＭ ＴｒｕｓｔＺｏｎｅ、ＡＭＤのＳｅｃｕｒｅ Ｅｎｃｒｙｐｔｅｄ Ｖｉｒｔｕａｌｉｚａｔｉｏｎ（ＳＥＶ）によって提供されるセキュアに隔離された実行のためのアーキテクチャと、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）に基づいたアーキテクチャとの間の差を抽象化で取り去り得る。ＡＰＩは、抽象化されたエンクレーブアーキテクチャのいくつかの詳細事項を抽象化するエンクレーブプラットフォームのためのインターフェースを含む。これらのエンクレーブプラットフォームインターフェースは、エンクレーブホストＡＰＩ、エンクレーブプラットフォームＡＰＩ、およびリモートアテステーションＡＰＩを含む。エンクレーブホストＡＰＩは、エンクレーブのライフサイクルを管理するため、ならびにエンクレーブへの通信およびエンクレーブからの通信を提供するために、信頼できないホストプロセスによって使用され得る。エンクレーブプラットフォームＡＰＩは、信頼できるエンクレーブプラットフォームによってエンクレーブに提供され得、アテステーション、密封、およびエンクレーブコンピューターをホストするコンピューター上で実行する信頼できないコードとの通信のためのセキュリティプリミティブ、ならびにメモリ管理およびスレッドスケジューリングなどのコアランタイムサポートを含み得る。リモートアテステーションＡＰＩは、エンクレーブおよびそのクライアントが同じコンピューター上でホストされない場合に、リモートアテステーションを実施するために使用され得る。例えば、リモートアテステーションＡＰＩは、データがリモートコンピューター上のエンクレーブプラットフォームによって提供される隔離下で実行する特定のアイデンティティを有するエンクレーブから生じた（またはそこから送信された）ことを検証するためにローカルクライアントによって使用され得る。より一般的には、リモートアテステーションＡＰＩは、ローカルクライアントとリモートエンクレーブとの間のセキュア通信チャネルを確立するために使用され得る。

【００１４】
[0035]エンクレーブは、一般に、コンピューター技術の領域に特有の問題、およびコンピューター技術の領域から生じる問題に対するソリューションを提供する。より具体的に

は、エンクレーブは、信頼できるコードセグメントおよび信頼できないコードセグメントの両方が単一のコンピュータープロセッサーのアドレス空間内に存在する場合に、信頼できるコードを信頼できないコードから分離するための機序を提供する。例えば、エンクレーブは、極秘またはプライベートデータにアクセスしなければならないコードと同じ汎用コンピューター上で実行する潜在的に信頼できないコード（バグまたはウイルスのいずれかを潜在的に含むコードなど）の問題に対してセキュリティソリューションを提供する。本開示の実施形態は、単一のエンクレーブまたはエンクレーブクライアントが複数のネイティブエンクレーブプラットフォームにより著作されることを可能にすることによってソフトウェア開発を簡略化すること、特定のコンピューターの特有のハードウェア特徴にカスタマイズされなければならないソフトウェアコンポーネントの数を減少させることによって企業のコンピューター管理を簡略化すること、ならびに、複数のコンピューター上でホストされるエンクレーブにわたってセキュアなエンクレーブ処理を分散させることなどの、分散データ密封を用いた新たなセキュアコンピューティングシナリオを可能にすることを含め、コンピューター技術の領域から生じるそのようなセキュリティ問題に対してさらに改善されたソリューションを提供する。

【００１５】

[0036]図１は、いくつかの信頼関係と一緒にエンクレーブシステムのハイレベルブロック図を描写する。エンクレーブシステム１００は、コード１８０およびデータ１８２を含むメモリのセキュアな隔離領域を含むエンクレーブ１７６（代替的にエンクレーブコンテナまたはセキュア実行環境と呼ばれる）を含む。コード１８０は、公開またはプライベートであり得、データ１８２は、公開またはプライベートであり得る。例えば、プライベートデータまたはコードは、データ所有者１４２に属し得る（または、プライベートであり得る）一方、公開データまたはコードは、ソフトウェアプロバイダー１４８などの別のパーティに提供されている場合がある。１つの実施形態において、エンクレーブコンテナ１７６内で実行するコードは、完全に公開であってプライベートではない場合がある一方、公開エンクレーブコードが入力として使用するか、または出力としてもたらすデータは、プライベートであり得る。別の実施形態において、コードがプライベートである一方でデータが公開である場合には、逆のことが可能である。さらに別の実施形態において、コードおよび入力データの両方が公開であり得る一方、入力データを用いてコードを実行する出力は、プライベートであり得る。コード、入力データ、および出力データの他の公開およびプライベートの組合せが可能である。

【００１６】

[0037]エンクレーブ１７６コンテナは、信頼できるハードウェア１７２上でホストされ、このハードウェア１７２は、信頼できないソフトウェア１７４を同時にホストし得る。エンクレーブシステム１００の主な目的は、コード１８０の完全性を維持すること、コード１８０の機密性を維持すること、データ１８２の完全性を維持すること、およびデータ１８２の機密性を維持することを含むリストから選択される少なくとも１つの態様を含み得る。信頼できないソフトウェア１７４（例えば、信頼できないソフトウェアへの開示、信頼できないソフトウェアによる修正、または同様のこと）からエンクレーブ１７６のコンテンツを保護することが目標である。信頼できるハードウェアは、製造業者１６２によって構築され、インフラ所有者１５２によって所有および管理される。

【００１７】

[0038]エンクレーブクライアント１０４は、エンクレーブ１７６がコード１８０およびデータ１８２を用いてその計算を実施するエンクレーブコンテナの外側のプロセスまたはプログラムであり得る。ローカルエンクレーブ実施形態において、エンクレーブクライアント１０４はまた、信頼できるハードウェア１７２上で実行していてもよい。リモートエンクレーブ実施形態において、エンクレーブクライアントは、１つのコンピューター上で実行し得る一方、信頼できるハードウェア１７２は、ネットワークを介してエンクレーブクライアントのコンピューターに接続される異なるリモートコンピューターである。ローカルエンクレーブの場合、エンクレーブクライアントプロセスはまた、エンクレーブクラ

イアントプロセスがローカルエンクレーブ１７６の作成を管理し得るという点において、エンクレーブコンテナ１７６のエンクレーブホストプロセスであり得る。リモートエンクレーブの場合、エンクレーブ１７６は、例えば、インフラ所有者１５２がクラウドコンピューティングサービスプロバイダーである場合、インターネットクラウドコンピューター上で実行され得、クラウドコンピューターは、製造業者１６２によって製造される信頼できるハードウェア１７２を含む。

【００１８】

[0039]エンクレーブクライアント１０４は、リクエストされた計算をエンクレーブ１７６によってセットアップするセットアップ１０６方法を含み得る。セットアップ１０６方法は、エンクレーブ１７６のセキュアコンテナの作成を引き起こすことと、バイナリコードをセキュアコンテナ内へコピーすることを含み得る、エンクレーブのインスタンス化を引き起こすこと（例えば、エンクレーブのインスタンス化を要求するために、信頼できないソフトウェア１７４にリクエストを送信することによって）と、例えばセキュアコンテナ内へコピーされたコード内の方法を呼び出すことによって、エンクレーブ内の計算を引き起こすことまたはリクエストすることとを含み得る。リクエストされた計算は、コード１８０を実行することを含み得、データ１８２は、リクエストされた計算に入力され得るか、またはリクエストされた計算の結果であり得る。リクエストされた計算に入力されるデータは、エンクレーブの外側にあるときには暗号化され得、暗号化された入力データは、エンクレーブの内側で使用される前に復号され得る。エンクレーブ１７６がリクエストされたタスクを完了すると、このタスクの結果を表すデータが、暗号化され、エンクレーブクライアント１０４へ送り返される。エンクレーブクライアント１０４が暗号化された結果を受信すると、検証１０８方法は、受信した結果の完全性および新鮮性を確認し得る。単一のソフトウェアプロバイダー１４８が、エンクレーブ１７６の内側で実行するためのコード１８０、およびエンクレーブクライアント１０４の部分として実行する検証１０８方法の少なくとも一部分の両方を提供し得る。

【００１９】

[0040]データ１８２のプライベート部分およびコード１８０のプライベート部分のプライバシーに対するデータ所有者の確信、ならびにエンクレーブ１７６によってもたらされる結果の正確さに対する確信は、信頼関係に基づき得る。例えば、データ所有者１４２は、エンクレーブコンテナ自体の中では実行していない場合があるエンクレーブクライアント１０４を信頼し得る。データ所有者は、さらに、エンクレーブ自体のソフトウェアプロバイダー１４８を信頼し得る。また、データ所有者は、信頼できるハードウェア１７２の製造業者を信頼し得る。信頼できるハードウェア１７２は、使用されるエンクレーブアーキテクチャに応じて多くの形態をとり得、ハードウェアセキュリティモジュール（ＨＳＭ）を含み得、この場合ＨＳＭは、例えば、トラステッドプラットフォームモジュール（ＴＰＭ）規格に準拠する。信頼できるハードウェア１７２は、例えば、ＴＰＭを含み得、そうでない場合はハードウェアのみを備え得る。例えば、ＭｉｃｒｏｓｏｆｔのＶＳＭエンクレーブアーキテクチャを使用した信頼できるハードウェア１７２の実装形態は、オペレーティングシステム仮想化命令のための命令を有するコモディティプロセッサーおよびＴＰＭを含み得る。ＭｉｃｒｏｓｏｆｔのＶＳＭエンクレーブアーキテクチャは、ゲストパーティションを管理するためのハイパーバイザー（仮想プロセッサー）を含み得、ハイパーバイザーは、ハイパーコールインターフェースをゲストパーティションに露出して、ゲストパーティションがハイパーバイザーと相互作用することを可能にし得る。ＭｉｃｒｏｓｏｆｔのＶＳＭアーキテクチャ内のエンクレーブコンテナは、特別なタイプのゲストパーティションとして実装され得る。ＩｎｔｅｌのＳＧＸエンクレーブアーキテクチャを有する信頼できるハードウェア１７２の例は、特別なエンクレーブ特有の命令およびセキュリティ機能性を有するプロセッサーを含み得る。

【００２０】

[0041]エンクレーブ１７６などのエンクレーブは、保護された領域からの読み込み、書き込み、または実行に対する制限をメモリの領域に提供することによってコード１８０お

よびデータ１８２などのコードまたはデータを保護し得る隔離された実行環境を提供し得る。この保護されたメモリ領域は、機密コードおよびデータのためのセキュアコンテナである。エンクレーブの保護されたメモリ領域からの実行に対する制限は、エンクレーブの外側のコードからエンクレーブの内側のコードへの、およびその逆の間での呼び出しまたは飛び越し命令などの実行転送に対する制限を含み得る。異なる制限が、エンクレーブの外側からのエンクレーブへのコールイン、およびエンクレーブの内側からのエンクレーブのコールアウトの間で履行され得る。エンクレーブの内側と外側との間のこれらの実行転送の履行は、ハードウェアによって、例えば、コモディティ仮想化ハードウェア技術を用いて、またはＩＮＴＥＬのＳＧＸプラットフォームなどの特殊ハードウェアを用いて、提供され得る。

【００２１】

[0042]図２は、機密性保証付きのメッセージ引き渡しのための例示的なプロセス２００を描写する。機密性保証は、本例ではアン２１０およびベン２３０などの２つのパーティ間の通信が、メッセージがネットワーク２１８などの公開または非保護通信媒体を通じて引き渡されるときに第三者から隠されたままであるというある程度の確証を提供する。本例では、アン２１２は、ベン２３０に機密メッセージを送信したいと思っている。機密データを含むメッセージブロック２１２は、暗号化２１４動作によって公開鍵２１６を使用して暗号化される。暗号化２１４動作は、例えば、Ａｄｖａｎｃｅｄ Ｅｎｃｒｙｐｔｉｏｎ Ｓｔａｎｄａｒｄ Ｇａｌｏｉｓ／Ｃｏｕｎｔｅｒ Ｍｏｄｅ（ＡＥＳ－ＣＧＭ）であり得るが、デジタル暗号化の当業者に知られている任意の暗号化動作であってもよい。暗号化されたブロック２２０は、ネットワーク２１８を通じてベンへと渡り得、ここで、暗号化されたブロック２３４は、ベンのために暗号化されていないメッセージブロック２３２をもたらすためにプライベート鍵２３６を用いて復号される。鍵および暗号化アルゴリズムの慎重な選択により、コンピューターデータ暗号化の分野において知られているように、メッセージは、公開ネットワークを通過するときにさえ機密のままであることができる。

【００２２】

[0043]図３は、完全性保証付きのメッセージ引き渡しのための例示的なプロセス３００を描写する。完全性保証は、本例ではアン３１０およびベン３５０などの２つのパーティ間の通信が、メッセージがネットワーク３４０などの公開または非保護通信媒体を通じて引き渡されるときに改ざんされない、または別のやり方で変更されないというある程度の確証を提供する。図３の例において、アン３１０は、メッセージ３１４が、ベン３５０がそれを受信するときに改ざんされない、または別のやり方で破損されないという自信があるように、メッセージ３１４をベン３５０に送信する。この完全性保証を提供するため、セキュアハッシング３１６プロセスがメッセージ３１４に作用して、ハッシュ値３１８をもたらす。次いで、ハッシュ値３１８は、署名３４２を生成するためにアンのプライベート鍵を使用した署名３２０プロセスによって署名される。署名３４２は、メッセージ３４４であるメッセージ３１４のコピーと一緒に公開ネットワーク３４０または他の非保護通信プロセスで送信され得る。次いで、ベンはメッセージ３５４を受信し、このメッセージ３５４について、ベンは、メッセージ３５４が、信頼できないネットワーク３４０を通過した後に、アンが送信したメッセージ３１４と同じであるという自信を有することができるように、完全性を検証することを望む。完全性を検証するため、受信したメッセージ３５４は、セキュアハッシング３１６と同一であるセキュアハッシング３５６によって処理されて、ハッシュ値３５８をもたらす。受信した署名３４２は、アンの公開鍵を使用した署名検証３６０プロセスによって検証される。次いで、署名３４２から抽出されるハッシュ値３１８は、署名が同じであることを検証する３８０ためにハッシュ値３５８と比較される。それらが同じである場合、メッセージは完全性を有するものとして承認される３８４。代替的に、メッセージ３１４がメッセージ３５４として受信される前に何らかのやり方で変更された場合、署名は正しくなく、メッセージは拒絶されることになる３８８。

【００２３】

[0044]いくつかの実施形態において、セキュアハッシング３１６およびセキュアハッシング３５６は、暗号学的ハッシュ関数であり得る。暗号学的ハッシュ関数は、任意サイズのデータを固定サイズのビットストリング（典型的には、はるかに小さい）にマッピングする一方向関数である。ハッシュ関数の出力は、ハッシュ値または単にハッシュと呼ばれ得る。優れたハッシュ関数は、ハッシュ出力のみを前提として任意サイズの入力を決定するのが困難であるという点において一方向である。優れたハッシュ関数では、入力における小さな変化でさえ出力における変化をもたらす。

【００２４】

[0045]通信システムは、機密性および完全性保証を組み合わせることができる。例えば、図２のメッセージブロック暗号化は、図３のメッセージハッシュの署名と組み合わされ得る。組合せシステムは、１つは送信者用、１つは受信者用の、公開／プライベート鍵ペアの２つのセットを必要とし得る。組合せシステムは、メッセージを復号するために受信者側で１つのプライベート鍵を使用し得る一方で（図２のようにベンのプライベート鍵）、メッセージハッシュに署名するために別のプライベート鍵を使用する（図３のようにアンのプライベート鍵）。

【００２５】

[0046]図４は、新鮮性保証付きのメッセージ引き渡しのための例示的なプロセス４００を描写する。新鮮性保証は、複数のメッセージが、本例ではアン４１０からベン４５０へなど、１つのパーティから別のパーティへ送信されるとき、受信者側で受信されるメッセージが直近のメッセージであるというある程度の確証を提供する。新鮮性保証は、完全性保証を基に構築され得、リプレイアタックを防ぐことができる。完全性保証により、悪意を持った第三者が、独自のメッセージを作成して、ベンがそのメッセージはアンによって作成されたものであると理解するように、それをベンに送信することは困難である。しかしながら、第三者は、アンによって実際に作成され、それが公開ネットワークを介して送信された際におそらく観察されたメッセージを取り込むことができ、第三者は、それを後になって別のときにベンに送信し得る（すなわち、メッセージをリプレイする）。ベンは、そのメッセージが実際にアンによって作成されたことを決定する（そうであったため）が、ベンは、アンがそのときにそれを送信した人物ではないことは知らない。これは、第三者によるベンまたはアンに対するリプレイアタックと呼ばれ得る。図４は、ノンスおよびタイムスタンプを使用して新鮮性保証を提供することによってリプレイアタックを防ぐための例示的なソリューションである。ノンスは、１回限りの乱数であり、同じ数字がノンスとして決して２回以上使用されないことを確実にするためのシステムと連動される。いくつかのシステムにおいて、ノンスは、使用されるすべての数字がそれより前に出たすべての数字よりも大きくなるように、単に単調増加する数字であり得る。

【００２６】

[0047]図４では、アンのメッセージ４１４は、ノンス４３４およびタイムスタンプ４３２と一緒にメッセージ４３６として公開ネットワーク４３０を介して送信され得る。ノンス４３４は、暗号学的にセキュアな擬似ランダム数発生器（ＣＳＰＲＮＧ）４２６によって生成され、タイムスタンプ４３２は、同期化クロック４２４によってもたらされる。デジタル暗号法の当業者に知られているＣＳＰＲＮＧシステムは数多く存在する。ネットワーク４３０のアン側の同期化クロック４２４は、ネットワークのベン側の同期化クロック４８０と同期される。ベン側では、メッセージ４５４が受信されると、付随するノンス４３４が、近時ノンス４７０のキャッシュ内に格納される。受信したメッセージ４５０の新鮮性は、２つの試験により検証され得る。まず、ノンス４３４は、現在受信したノンス４３４が前に見たものであるかどうかを決定するために、ノンス４３４を近時ノンス４７０のキャッシュと比較することによって、ボックス４６０において試験される。受信したノンス４３４が前に見たものである場合、メッセージ４５４は、ボックス４６８においてリプレイメッセージとして拒絶される。受信したノンス４３４が前に見たものでない場合、メッセージは、この第１の試験では、ＯＫであることがボックス４６４において決定される。次に、受信したタイムスタンプ４３２が、ローカル同期化クロック４９０と比較され

る。タイムスタンプが近時である場合、メッセージ４５４は、ボックス４９４において容認できることが決定され、そうでない場合、メッセージ４５４は、ボックス４９８において期限切れとして拒絶される。近時タイムスタンプがボックス４９０において近時であるかどうかを決定するときに許容される遅延の量は、同期化クロック４２４と同期化クロック４８０との間の予期されるクロックスキューならびにメッセージ処理およびネットワーク４３０を介した伝送における時間遅延に依存し得る。

【００２７】

[0048]通信システムは、新鮮性保証を、図２のような機密性保証および図３のような完全性保証のいずれかまたは両方と組み合わせることができる。３つすべてを組み合わせるシステムにおいて、ネットワーク４３０を介して送信されるメッセージ４３６は、アンの元のメッセージ４１４の暗号化バージョンであり、メッセージ４１４の署名付きハッシュを含む署名が、タイムスタンプ４３２、ノンス４３４、およびメッセージ４３６と一緒に含まれる。

【００２８】

[0049]図５は、エンクレーブのソフトウェアアテステーションのための例示的なプロセス５００を描写する。ソフトウェアアテステーションは、図６のものなどの鍵合意プロトコルと組み合わされるとき、それが、信頼できるハードウェアによって作成された隔離されたコンテナの内側でホストされる特定のソフトウェアにより共有秘密を確立したということを検証者に保証することができる。図５の実施形態において、エンクレーブクライアント５１０（アテステーション検証者）は、トラステッドプラットフォーム５３０上でエンクレーブのセキュアな計算サービスを使用することを望み得る。トラステッドプラットフォーム５３０は、コンピューター（描写されない）上でホストされ、その結果、トラステッドプラットフォーム５３０は、ホスティングコンピューターのサブセットを備え得る。トラステッドプラットフォーム５３０は、ホスティングコンピューターのハードウェア要素およびソフトウェア要素を備え得る。エンクレーブは、セキュアエンクレーブコンテナ５３６、ならびにその内側のコードおよびデータ、例えば、公開コードおよびデータ５３８ならびに秘密コードおよびデータ５４２を含む。

【００２９】

[0050]３つのプロセスが、例示的なプロセス５００において組み合わされ、共有鍵ＳＫをもたらす鍵交換プロセス、トラステッドプラットフォーム５３０上のエンクレーブのエンクレーブクライアント５１０へのアテステーションのためのアテステーションプロセス、およびセキュアな計算が行われる。第１のプロセスからの共有鍵ＳＫは、セキュアな計算の入力および出力を通信するために使用される。鍵交換において、エンクレーブクライアントは、例えば、図６のディフィー・ヘルマン鍵交換（ＤＫＥ）プロトコルについて以下に説明されるように、エンクレーブクライアントのプライベート鍵Ａおよびジェネレータ関数ｇから、ボックス５１２内に格納される$g^A$をコンピューティングする。コンピューティングされた$g^A$は、次いで、メッセージ５２０内でトラステッドプラットフォーム５３０に送信される。メッセージ５２０は、暗号化なしに、およびアテステーションが完了する前に、安全に送信され得る。セキュアエンクレーブコンテナ５３６の内側のソフトウェアは、同じジェネレータ関数ｇを使用して$g^B$をコンピューティングするためにエンクレーブプライベートＢを使用し得る。Ｂおよび$g^B$の両方は、ボックス５４０においてエンクレーブコンテナに格納され得る。

【００３０】

[0051]エンクレーブのアイデンティティを証明するために（どのコードがセキュアエンクレーブコンテナ５３６の内側で実行しているかに関して確証を提供するために）、アテステーションメッセージ５２２が、エンクレーブクライアント５１０に送信される。アテステーションメッセージは、図３のような完全性について署名された特別なメッセージであり得る。特別なメッセージは、エンクレーブに関するアイデンティティ情報を含み得る。図５の実施形態にあるようにＤＫＥと組み合わされるとき、特別なメッセージは、鍵交換パラメーターも含み得る。図５の実施形態において、セキュアエンクレーブコンテナ５

３６の、公開コードおよび公開データの初期状態５３８が、エンクレーブアイデンティティとして使用されるが、他のアイデンティティも可能である。初期状態５３８全体をアテステーションメッセージ内で送信する代わりに、初期状態のハッシュ、Ｍ＝Ｈａｓｈ（初期状態）が代わりに送信される。アテステーションメッセージ５２２は、メッセージコンテンツ（ＭおよびｇＢ）、およびメッセージコンテンツの署名（ＳｉｇｎＡＫ（Ｈａｓｈ（ｇＢ），Ｍ））を含む。メッセージコンテンツの署名は、例えば、セキュアエンクレーブコンテナ５３６の内側のソフトウェアが、エンクレーブをホストするトラステッドプラットフォーム５３０に、コンピューティングされたｇＢのハッシュおよびエンクレーブのアイデンティティを証明することを要求することによって、作成され得る。トラステッドプラットフォーム５３０は、ＳｉｇｎＡＫ（Ｈａｓｈ（ｇＢ），Ｍ）をもたらすためにプラットフォームアテステーション鍵（ＡＫ）５３２を使用して署名を提供することによって、これを行い得る。本例では、エンクレーブアイデンティティは、初期状態５３８のハッシュＭであるが、他のアイデンティティステートメントが可能である。このアテステーション署名ＳｉｇｎＡＫ（Ｈａｓｈ（ｇＢ），Ｍ）は、値ｇＢをエンクレーブアイデンティティＭに結合し、また、ｇＢおよびＭの両方をトラステッドプラットフォーム５３０に結合する。エンクレーブクライアント５１０は、次いで、アテステーション署名およびエンクレーブアイデンティティを検証することによって、アテステーションメッセージを検証することができる。署名は、図３にあるように、アテステーション鍵ＡＫに対応する公開鍵を使用して検証され得る。署名を検証することが、アテステーションメッセージ内のエンクレーブアイデンティティの完全性保証を提供し得る。エンクレーブアイデンティティは、アテステーションメッセージ内のアイデンティティ情報を、独立して知られているアイデンティティ値と比較することによって検証され得る。例えば、アテステーションメッセージ内のアイデンティティ情報が、エンクレーブの初期状態のハッシュである場合、エンクレーブクライアント５１０は、初期状態のハッシュを知り得るか、または既知の初期状態からそのようなハッシュをコンピューティングすることができる場合があり、この値は、次いで、アテステーションメッセージ内で提供されるアイデンティティ値と比較され得る。

【００３１】

[0052]共有鍵ＳＫをもたらすために、エンクレーブクライアント５１０およびセキュアコンテナ５３６の内側のコードの両方が、共有鍵ＳＫとしての役割を果たし得るｇＡＢ（Ａ×Ｂの積に適用されるジェネレータ関数ｇ）を生成することができる。共有鍵ＳＫは、エンクレーブクライアント５１０とエンクレーブとの間の機密性のためにメッセージを暗号化するために、例えば、エンクレーブコンテナ５３６の内側のコードへ入力データを送信するため、およびエンクレーブコンテナ５３６の内側のコードから出力データを送信するために使用される。共有鍵は、エンクレーブクライアントまたはエンクレーブのいずれかが他方のプライベート鍵を知ることなく、ボックス５４０および５１４において通信チャネルの各側で独立してもたらされる。例えば、図５の実施形態において、秘密コードおよびデータは、それをメッセージ５２４内でトラステッドプラットフォーム５３０に送信する前に、以前に確立された共有鍵ＳＫを用いて秘密コードおよびデータを暗号化し、ＥｎｃＳＫ（秘密コード／データ）をもたらすことによって、エンクレーブクライアント５１０によってセキュアに提供され得る。他の実施形態において、セキュアエンクレーブコンテナ５３６内で実行されるか、またはセキュアエンクレーブコンテナ５３６によって使用される秘密コードおよびデータ５４２は、他の場所から生じ得る。安全な計算は、計算結果５４４をもたらすために、秘密コードおよび／またはデータ５４２を使用してセキュアエンクレーブコンテナ５３６の内側で実施され得る。計算結果５１６は、次いで、それらをメッセージ５２６内でエンクレーブクライアントに送信する前に共有鍵ＳＫ（ＥｎｃＳＫ（ｒｅｓｕｌｔｓ））を用いて結果を暗号化することによって、エンクレーブクライアント５１０へとセキュアに通信されて戻され得る。

【００３２】

[0053]上に説明される図５のプロセスは、エンクレーブクライアントが特定のアイデン

ティティの「本物の」エンクレーブと通信している、およびエンクレーブがエンクレーブプラットフォームによって保護されるという、エンクレーブクライアントに対する保証を提供する。これは、通信チャネルの反対側にあるエンティティに関してエンクレーブコンテナの内側のコードに対するいかなる保証も提供しない。代替的な実施形態において（描写されない）、エンクレーブクライアントに関するそのような保証は、エンクレーブ自体としてエンクレーブクライアントを実行することによって提供され得る。この代替的な実施形態において、エンクレーブクライアントは、信頼できるエンクレーブプラットフォームに$g^A$のハッシュへの署名を求める場合があり、これは次いで、他のエンクレーブによって検証され得る。

【００３３】

[0054]アテステーションは、ローカルまたはリモートで行われ得る。図５では、エンクレーブクライアント５１０は、トラステッドプラットフォーム５３０として同じコンピューター上に存在する場合とそうでない場合とがあるため、その結果として、エンクレーブクライアント５１０とトラステッドプラットフォーム５３０との間の通信は、単一のコンピューター内で発生し得るか（例えば、同じコンピューター上の異なるプロセス間でデータバッファを引き渡すことによって）、またはエンクレーブクライアント５１０をトラステッドプラットフォーム５３０に接続するコンピューターネットワークを介して発生し得る。ローカルアテステーションは、エンクレーブクライアント５１０およびトラステッドプラットフォーム５３０が同じローカルコンピューター上でホストされるときに実施され得る。ローカルアテステーションの中間生成物または結果は、アテステーションレポートと呼ばれ、図５の例では、$Sign_{AK}$（Ｈａｓｈ（$g^A$），Ｍ）である。リモートアテステーションは、エンクレーブクライアント５１０およびトラステッドプラットフォーム５３０が異なるコンピューター上でホストされるときに発生し得る。リモートアテステーションの中間生成物または結果は、アテステーションクォートと呼ばれ、いくつかの場合において、これは、ローカルアテステーションレポートと非常に類似し得るか、または同一であり得る。他の場合において、アテステーションクォートは、クォートを提供するコンピューターまたはネイティブプラットフォームに関連する追加の信頼の中間生成物を含み得る。そのような追加の信頼の中間生成物は、ＴＰＭと関連付けられたＴＣＧログなどのホスト健全性診断書を含み得る。エンクレーブのアテステーションは、エンクレーブのアイデンティティの任意の層上で実施され得る。エンクレーブは、ネスト化または階層アイデンティティを有し得、より高いレベルのアイデンティティへのアテステーションは、インスタンス化されたエンクレーブが、アイデンティティレベルが増大するにつれて潜在的なエンクレーブインスタンス化の漸進的により大きいグループのメンバーであるということを証明し得る。より高いレベルは、より低いレベルの潜在的なエンクレーブインスタンス化の上位集合に対応し得る。アイデンティティの例示的な階層は、最も低い最も具体的なレベルからより高いあまり具体的ではないレベルまで含み得、ＥｘａｃｔＨａｓｈ、ＩｎｓｔａｎｃｅＨａｓｈ、ＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、およびＡｕｔｈｏｒＩＤであり得る。

【００３４】

[0055]エンクレーブのアイデンティティは、エンクレーブのセキュアコンテナ内へロードされるバイナリファイル（エンクレーブバイナリ）から導出され得る。エンクレーブバイナリは、そのオーサーによって、オーサーのプライベート鍵を使用して署名され得る。ＥｘａｃｔＨａｓｈレベルアテステーションでは、アテステーションレポート（アテステーションレポートをもたらすためのハッシュ関数への入力）をコンピューティングするために使用される初期状態５３８は、トラステッドプラットフォーム５３０と関連付けられたバイナリを除き、セキュアコンテナ５３６内へロードされる全バイナリファイルのコンテンツ全体を含み得る。

【００３５】

[0056]ＩｎｓｔａｎｃｅＨａｓｈアイデンティティにおけるアテステーションは、初期状態５３８のサブセットを含み得る。サブセットは、セキュアコンテナ５３６内へロード

されるエンクレーブバイナリファイル（バイナリファイル）が、それらのエンクレーブバイナリファイルのオーサーによって当初に署名されたときに特定され得る。例えば、第１の（または一次）エンクレーブバイナリファイルは、第１のエンクレーブバイナリファイルが依存する他のエンクレーブバイナリファイルのアイデンティティのリストを含み得る。リストされる各アイデンティティについては、リストされる各バイナリファイルがＩｎｓｔａｎｃｅＨａｓｈアテステーションレポートをもたらすためにハッシュ関数によって測定されるか否かを示すためにフラグが第１のバイナリファイル内に含まれ得る。

【００３６】

[0057]より高いレベルのエンクレーブＩＤへのアテステーションは、任意のエンクレーブバイナリのコンテンツ全体をハッシュ関数にかけることを含まない場合がある。代わりに、ＩＤのデータ構造のみが、ハッシュ関数にかけられ得る。例えば、エンクレーブバイナリファイルは、その特定のエンクレーブバイナリファイルに固有の画像ＩＤ（ＩｍａｇｅＩＤ）、その特定のエンクレーブバイナリファイルを含み、かつ同じオーサーによって著作されるエンクレーブバイナリファイルのグループに固有のファミリーＩＤ（ＦａｍｉｌｙＩＤ）、および同じオーサーによってすべて著作されるエンクレーブバイナリファイルのファミリーのグループに固有のオーサーＩＤ（ＡｕｔｈｏｒＩＤ）を示す、汎用一意識別子（ＵＵＩＤ）などの、より高いレベルのエンクレーブ識別子のリストを含み得る。ＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、およびＡｕｔｈｏｒＩＤは、バイナリが当初に署名されるときにエンクレーブバイナリのオーサーによって指定され得る。エンクレーブクライアントがエンクレーブバイナリにアクセスして、オーサーの公開鍵（またはオーサーと関連付けられた公開鍵）を使用してそれらのバイナリ上のオーサーの署名を検証することができる場合には、エンクレーブアイデンティティなりすましは防がれ得る。これは、オーサーによって指定される任意のより高いレベルのアイデンティティを含め、エンクレーブバイナリの完全性を、そのエンクレーブオーサーによって作成されていたと検証する。

【００３７】

[0058]図６は、例示的なディフィー・ヘルマン鍵交換（ＤＫＥ）プロトコル６００を描写する。ＤＫＥは、完全性保証のみを有する通信チャネルにわたって共有鍵Ｋを確立するための１つの例示的なプロセスであり、デジタル暗号法の分野で知られている共有鍵を作成するための他のプロセスが使用されてもよい。図６のＤＫＥ例において、秘密鍵Ｋは、アン６１０とベン６５０との間で公開（非セキュアの）通信媒体を介して決して明示的にＫを通信することなくアン６１０とベン６５０との間で共有される。プロセスが開始する前に、１）大素数ｐおよび２）Ｚｐ内のジェネレータｇが確立され得、アンおよびベンの両方に知られ得る。次いで、プロセスは、アンおよびベンの両方が１からｐの間で乱数を選択することにより開始する。ボックス６１２において選択されるアンの乱数はＡであり、ボックス６５２において選択されるベンの乱数はＢである。アンは、ボックス６１４において自分の乱数を使用して$g^{A} \mod p$をコンピューティングし、ボックス６１６においてその量を伝送し、これはボックス６５６においてベンにより受信される。ベンは、ボックス６５４において自分の乱数を使用して$g^{B} \mod p$をコンピューティングし、これは、ボックス６５６においてアンに伝送され、ボックス６１８において受信される。アンは、ボックス６２０において、共有鍵Ｋを$(g^{B} \mod p)^{A} = g^{AB} \mod p$としてもたらすことができ、ベンは、ボックス６６０において、共有鍵Ｋを$(g^{A} \mod p)^{B} = g^{AB} \mod p$としてもたらすことができる。中間者攻撃を防ぐために、ボックス６１６および６５８からの非保護ネットワークを介したアンとベンとの間の通信は、例えば図３のものなどのプロセスを使用して作成されるメッセージ完全性保証を含み得る。

【００３８】

[0059]図７は、ソフトウェアアテステーションのための例示的な信頼チェーン７００を描写する。ソフトウェアアテステーションにおける信頼チェーンは、図５のトラステッドプラットフォーム５３０などのトラステッドプラットフォームの製造業者によって所有される署名鍵に根差し得る。トラステッドプラットフォームは、セキュアプロセッサーまた

はハードウェアセキュリティモジュール（ＨＳＭ）などのハードウェアコンポーネントを含み得、したがって製造業者は、コンピューターハードウェアのプロバイダーであり得、また、トラステッドプラットフォームのためのソフトウェアを提供し得る。製造業者は、検証者７０２により信頼され得、検証者は、製造業者ルート鍵７３６の公開鍵ＰｕｂＲＫ７３２を知り得る。図５のエンクレーブクライアント５１０は、セキュアコンテナ７０８に関する確証を有することを望み得る検証者７０２の例である。製造業者は、認証局としての機能を果たし、アテステーション署名をもたらすために使用される固有のアテステーション鍵７２２と一緒に、製造業者が生産するトラステッドプラットフォームの各インスタンス、例えば各セキュアプロセッサーを提供し得る。製造業者はまた、各アテステーション鍵７２２のための承認証明書７２８を発行する。製造業者ルート鍵７３６は、承認証明書７２８に署名するために使用されるプライベート鍵ＰｒｉｖＲＫ７３４を含む。承認証明書の署名は、例えば図３に示されるような、完全性保証を提供する。

【００３９】

[0060]承認証明書７２８は、アテステーション鍵７２２の公開鍵ＰｕｂＡＫ７２４を含む。承認証明書７２８は、アテステーション鍵７２２がソフトウェアアテステーションのために使用されることを示し得、検証者７０２へ通信され得る。検証者は、セキュアコンテナ７０８のアテステーションを検証することを望む任意のエンティティであり、例えば、検証者７０２は、セキュアな計算がセキュアコンテナ７０８の内側で行われることを望む図５のエンクレーブクライアント５１０であり得る。検証者７０２は、完全性および承認証明書の発信元を検証するために、ＰｕｂＲＫ７３２を使用して承認証明書を調査し得る。検証者はまた、承認証明書からＰｕｂＡＫ７２４を抽出し得る。承認証明書は、アテステーション鍵７２２が、アテステーション署名をもたらすためだけに使用されること、およびアテステーション鍵７２２のプライベート鍵ＰｒｉｖＡＫ７２６が、耐タンパー性ハードウェア７３０の記憶装置内など、トラステッドプラットフォームの一般的にアクセス可能なコンピューターメモリとは分離される記憶装置内に排他的に維持されることを要求し得る証明ポリシーと関連付けられ得る。耐タンパー性ハードウェアは、例えば、トラステッドプラットフォームモジュール（ＴＰＭ）規格に準拠するハードウェアであり得る。

【００４０】

[0061]セキュアコンテナ７０８は、トラステッドプラットフォーム７３６上でインスタンス化され得る。セキュアコンテナ７０８のインスタンス化は、非セキュアの処理によるアクセスを制限されるセキュアコンテナのための隔離されたメモリ空間を規定することを含み得る。非セキュアの処理は、例えば、トラステッドプラットフォームの外側からではあるがトラステッドプラットフォームをホストするコンピューター上でのアクセス、またはトラステッドプラットフォームの内側の他のセキュアコンテナ内からのアクセスを含み得る。セキュアコンテナ７０８のインスタンス化はまた、公開コードおよびデータ、例えば、図５の初期状態５３５を、セキュアコンテナ内へロードすることを含み得る。

【００４１】

[0062]インスタンス化されたセキュアコンテナ７０８は、機密通信のための共有鍵を確立するために検証者７０２と鍵を交換することができる。鍵交換プロセスは、図５の鍵交換プロセスまたは図６のＤＫＥプロセスであり得る。検証者は、例えば図６のボックス６１６にあるように、鍵交換メッセージ１　７０４をトラステッドプラットフォーム７３６に送信し、トラステッドプラットフォーム７３６は、例えば図６ボックス６５８にあるように、鍵交換メッセージ２　７０６を検証者７０２に返送する。

【００４２】

[0063]アテステーション署名７１０は、セキュアコンテナ７０８がインスタンス化され、鍵交換が完了した後に作成され得る。インスタンス化されたセキュアコンテナ７０８は、セキュアコンテナのすべてまたは部分に対して暗号学的ハッシュ関数を実行することによって測定され得る。これは、隔離されたメモリのコンテンツ、および、隔離されたメモリ、セキュアコンテナのインスタンス化の間に使用もしくは影響されるトラステッドプラ

ットフォームと関連付けられた任意の他のメモリ、またはこれらの任意のサブセットもしくは部分内へロードされるバイナリファイルに対してハッシュ関数を実行することを含み得る。このハッシュ関数を実行する出力は、アテステーション署名７１０の部分である、測定値７１２である。鍵交換メッセージ７０４および７０６の暗号学的ハッシュも、データ７１４として描写されるアテステーション署名７１０と共に含まれ得る。測定値７１２およびデータ７１４は、アテステーションプライベート鍵ＰｒｉｖＡＫ７２６を使用して署名され得る。アテステーション署名は、次いで、測定値７１２およびデータ７１４と一緒に検証者７０２に送信され得る。検証者は、承認証明書からのＰｕｂＡＫ７２４を使用してアテステーション署名の完全性を検証することができ、ＰｕｂＡＫ７２４は、図７の例では、測定値７１２およびデータ７１４の完全性の検証も可能にする。検証者７０２は、測定値７１２を予測される結果（例えば、測定値７１２の同じハッシュをローカルで実施することによって決定される予測される結果）と比較することによってセキュアコンテナ７０８の完全性を検証することができ、また、（例えば、データ７１４のハッシュは、鍵交換メッセージ２　７０６に結び付けられるため）データ７１４を調査することによってアテステーション署名がこの特定の検証者７０２通信路インスタンスのために作成されたことを検証することができる。これらの検証動作および上の承認証明書の検証の後、検証者はこのとき、検証者が、確立された共有鍵を使用したセキュアコンテナ７０８との機密性および完全性の両方を有する通信を確立することができること、トラステッドプラットフォームハードウェアが、その製造業者に従って信頼され得ること、ならびにセキュアコンテナを作成するために使用されるトラステッドプラットフォームのソフトウェア状態が知られていることについて何らかの確証を有する。検証者７０２はこのとき、プライベートコードおよび／またはプライベートデータを使用してセキュアコンテナ７０８内でのセキュアな処理をリクエストする準備が整っている。

【００４３】

[0064]エンクレーブ抽象化プラットフォームおよびプリミティブ

[0065]図８は、例示的なローカルエンクレーブシステムのためのソフトウェアコンポーネントインターフェースのブロック図である。エンクレーブシステム８００は、エンクレーブ８１４およびエンクレーブのクライアント８１６をホストするネイティブエンクレーブプラットフォーム８１２を有するコンピューター８１０を含む。ネイティブプラットフォーム８１２は、例えば、ＩｎｔｅｌのＳＧＸまたはＭｉｃｒｏｓｏｆｔのＶＳＭに基づいたハードウェアおよび／またはソフトウェアコンポーネントであり得る。エンクレーブ８１０は、図１のエンクレーブ１７６であり得る。エンクレーブのためのネイティブプロトコル８４４は、エンクレーブ８１４、クライアント８１６、およびネイティブプラットフォーム８１２の間の通信のために使用され得る。図８に描写されるように、ネイティブプロトコル８４４は、エンクレーブ８１４内のインターフェース８２０、ネイティブプラットフォーム内のインターフェース８２２および８２４、ならびにクライアント内のインターフェース８２６を含む。これらのインターフェースは、ソフトウェアコンポーネント内のＡＰＩまたはＡＢＩであり得る。

【００４４】

[0066]これらのソフトウェアインターフェース８２０、８２２、８２４、および８２６の使用は、ソフトウェアコンポーネント間の実行制御転送を含み得る。制御転送は、制御が転送されているソフトウェアコンポーネント内のエントリーポイント（命令のアドレス）への呼び出しまたは飛び越し命令を実行することを含み得る。例えば、ネイティブプラットフォーム８１２がソフトウェアコンポーネントである場合、ネイティブプラットフォーム８１２からクライアント８１６への制御転送は、ネイティブプラットフォーム８１２内の呼び出しまたは飛び越し命令が、呼び出すまたは飛び越すべきクライアント８１６内のアドレスを特定して実行されるときにソフトウェアインターフェース８２６を介して発生し得る。クライアント８１６の内側の特定されたアドレスは、インターフェース８１６内の関数または方法のためのエントリーポイントであり得る。制御転送は、インターフェース８２０を介してネイティブプラットフォーム８１２からエンクレーブ８１４へ、イン

ターフェース８２２を介してエンクレーブ８１４からネイティブプラットフォーム８１２へ、インターフェース８２４を介してクライアント８１６からネイティブプラットフォーム８１２へ、およびインターフェース８２６を介してネイティブプラットフォーム８１２からクライアント８１６へ、図８内の矢印で示される。ネイティブプロトコル８４４は、インターフェース８２０、８２２、８２４、および８２６を介した通信のパターンを含み得る。

【００４５】

[0067]いくつかの実施形態において、ネイティブプラットフォーム８１２は、例えばエンクレーブを管理するための特別なプロセッサー命令により、ハードウェアコンポーネントとして少なくとも部分的に実施され得る。そのような特別なハードウェア命令は、ネイティブプラットフォーム８１２ソフトウェアコンポーネントの部分として実行され得る。代替的な実施形態において、ネイティブプラットフォーム８１２の関数のうちの一部またはすべてのためにソフトウェアコンポーネントが存在しない場合がある。これらの代替的な実施形態において、ネイティブプラットフォームインターフェース８２２および８２４は、ソフトウェアエントリーポイントの代わりにハードウェア命令であってもよいため、ネイティブプラットフォーム８１２の関数は、エンクレーブ８１４もしくはクライアント８１６によって使用され得るか、または、呼び出しもしくは飛び越し命令を実行する代わりに、特別なハードウェア命令を代わりにそれぞれエンクレーブ８１４もしくはクライアント８１６内で実行することによって使用され得る。

【００４６】

[0068]いくつかの実施形態において、エンクレーブ８１４のクライアント８１６は、それ自体がエンクレーブであり得る。例えば、エンクレーブクライアント８１６は、インターフェース８２４を使用して、エンクレーブ８１４が作成されることをリクエストし得る。これらの実施形態において、ネイティブプラットフォーム８１２を介したエンクレーブ８１４とクライアント８１６との間の通信は、実際には、２つのエンクレーブ同士の通信である。クライアント８１６もエンクレーブであるとき、エンクレーブクライアント８１６はまた、インターフェース８２２を使用し、８２０に類似するインターフェース（描写されない）を露出し得る。

【００４７】

[0069]図９は、抽象化層を有する例示的なローカルエンクレーブシステムのためのソフトウェアコンポーネントインターフェースのブロック図である。エンクレーブシステム９００は、ネイティブプロトコル９４４および抽象化プロトコル９４０、９４２間を翻訳するための抽象化プラットフォーム９１２を含む。ネイティブプラットフォーム９１８は、図８の抽象化プラットフォーム８１２に類似し得、インターフェース９２８および９３０は、図８のインターフェース８２０、８２２、８２４、および８２５の機能を組み合わせ得る。エンクレーブ抽象化プロトコル９４０は、エンクレーブ９１４のためのインターフェース９２０、９２２を含む一方、クライアント抽象化プロトコル９４２は、クライアント９１６のためのインターフェース９２４、９２６を含む。図８にあるように、コンピューター９１０上で実行するクライアント９１６は、インターフェース９２４を介してエンクレーブ９１４の作成をリクエストし得る。抽象化層９１２は、ネイティブプラットフォーム９１８と共にネイティブプロトコル９４４およびインターフェース９２８、９３０を使用してエンクレーブ９１４の作成を引き起こし得る。クライアント９１６およびエンクレーブ９１４は、ネイティブプラットフォーム９１８およびネイティブプロトコル９４４がＩｎｔｅｌ　ＳＧＸまたはＭｉｃｒｏｓｏｆｔ　ＶＳＭなどの異なるエンクレーブアーキテクチャに基づくとき、抽象化プロトコル９４０および９４２を使用し得る。図８にあるように、エンクレーブ９１４のクライアント９１６は、それ自体がエンクレーブであり得、ネイティブプラットフォーム９１８は、ハードウェアおよび／またはソフトウェアコンポーネントを含み得る。

【００４８】

[0070]エンクレーブ９１４およびクライアント９１６は、直接的に通信しなくてもよく

、代わりに抽象化プラットフォーム９１２を介してのみ通信し得る。直接通信は、例えばエンクレーブ９１４メモリの隔離に起因して、可能ではない、または望ましくない場合がある。エンクレーブメモリ隔離は、エンクレーブの隔離されたメモリからの読み込み、エンクレーブの隔離されたメモリへの書き込み、またはエンクレーブの隔離されたメモリ（内へのもしくはから外への飛び越し）実行を防ぎ得る。

【００４９】

[0071]エンクレーブ９１４は、コンピューター９１０のエンクレーブセキュアコンテナの内側に位置する命令を含み得る。クライアント９１６は、コンピューター９１０のメモリアドレス空間内に位置するが、エンクレーブ９１４のセキュアコンテナの外側に位置する命令を含み得る。抽象化プラットフォーム９１２は、エンクレーブ９１４のセキュアコンテナの内側または外側にある命令としてなど、様々なやり方で実施され得、また、ハイパーコール内から実行される命令を含み得る。抽象化プラットフォーム９１２がエンクレーブ９１４のセキュアコンテナ内に少なくとも部分的に含まれる場合、セキュアコンテナの内側の抽象化プラットフォームコードは、エンクレーブ９１４のコードの残りの部分とは別個に著作され得、公開ＡＰＩ／ＡＢＩを介して他のエンクレーブコードと相互作用するだけであり得る。そのような抽象化プラットフォームコードは、エンクレーブセキュアコンテナの内側のコードの残りの部分に静的にリンクされ得るか、または動的にリンクされ得る。静的にリンクされた抽象化プラットフォームコードは、抽象化プラットフォームと関連付けられ、エンクレーブ９１４により特有であるコードと一緒に、エンクレーブ９１４がインスタンス化され得るバイナリ画像内に含まれる（静的にリンクされる）オブジェクトコードであり得る。動的にリンクされた抽象化プラットフォームの場合、エンクレーブ９１４により特有であるエンクレーブコード、および抽象化プラットフォームとより一般的に関連付けられたコードは、別個のバイナリ画像から供給され得る。動的にリンクされた例については、図１４を参照されたい。

【００５０】

[0072]図１０は、抽象化層を有する例示的なリモートエンクレーブシステムのためのソフトウェアコンポーネントインターフェースのブロック図である。リモートエンクレーブシステム１０００は、コンピューター１０１０上のエンクレーブ１０１４、および別個のコンピューター１０５０上のエンクレーブ１０１４のクライアント１０５６を含む。クライアントスタブ１０１６および抽象化リモート処理プラットフォーム１０５２の組合せは、エンクレーブ１０１４とクライアント１０５６との間の相互作用を促進し得る。コンピューター１０１０内の多くの要素は、図９のコンピューター９１０の同一名の要素と同一であり得るか、または類似し得る。特に、抽象化プラットフォーム１０１２、プロトコル１０４０、１０４２、１０４４、およびネイティブプラットフォーム１０１８は、それぞれ対応する要素９１２、９４０、９４２、９４４、および９１８と類似し得るか、または同一であり得る。

【００５１】

[0073]クライアントスタブ１０１６は、ネットワーク通信１０８０を介して抽象化リモート処理プラットフォーム１０５２と通信し得る。リモートクライアントプロトコル１０８２、およびインターフェース１０６４、１０６６は、クライアント抽象化プロトコル１０４２、およびインターフェース１０２４、１０２６と類似し得る。しかしながら、リモートクライアントプロトコルは、リモート処理のための追加の機能性を含み得る。例えば、エンクレーブの作成をリクエストするためのＣｒｅａｔｅＥｎｃｌａｖｅなどのインターフェース１０６４内の方法は、追加的に、エンクレーブが作成されることがリクエストされる場合に、コンピューター１０１０などのエンクレーブホストコンピューターを特定する能力を含み得る。リモートクライアントプロトコルを介してクライアント１０５６に提供されるエンクレーブ１０１４のアテステーションクォートは、アテステーションレポートの代わりに、またはそれに加えて提供され得る。クライアント１０５６を有するコンピューター１０５０は、ネイティブエンクレーブプラットフォーム１０５８を含む場合とそうでない場合とがある。ネイティブプラットフォーム１０５８が存在する場合、ネイテ

ィブプラットフォーム１０５８は、サンプル・エンクレーブ・アーキテクチャ・ネイティブ・プラットフォーム１０１８に準拠する場合とそうでない場合とがあり、故に、ネイティブプロトコル１０４４およびリモートネイティブプロトコル１０８４は、同じではない場合がある。

【００５２】

[0074]代替的な実施形態において（描写されない）、クライアントスタブ１０１６は存在しない場合があり、抽象化プラットフォーム１０１２は、ネットワークを介して抽象化リモート処理プラットフォーム１０５２と直接的に通信し得る。

【００５３】

[0075]図９および図１０の９４０、９４２、１０４０、１０４２、１０８２などのエンクレーブ抽象化プロトコルは、Ｉｎｔｅｌ　ＳＧＸおよびＭｉｃｒｏｓｏｆｔ　ＶＳＭなどの複数のネイティブプラットフォーム上で構築されるエンクレーブを管理および使用するために様々なインターフェース方法またはエントリーポイントを含み得る。これらの方法は、複数のネイティブプラットフォーム上で実施され得るエンクレーブプリミティブを提供し得、故にネイティブプラットフォームの「抽象化」を提供する。ここに開示されるエンクレーブプリミティブは、エンクレーブライフサイクル管理、アテステーション、データ密封、制御転送、単調カウンター、および信頼できる時刻を含む。

【００５４】

[0076]エンクレーブライフサイクル管理のためのプリミティブは、エンクレーブ９１４などのエンクレーブのインスタンス化または終了を引き起こすための方法を含み得る。ライフサイクル管理プリミティブは、クライアント抽象化プロトコル９４２の一部であり得、およびより具体的には、クライアント９１６による使用のためのインターフェース９２４の部分として抽象化プラットフォーム９１２によって実施され得る。

【００５５】

[0077]エンクレーブをインスタンス化または作成するための方法は、セキュアエンクレーブコンテナの隔離されたメモリ内へロードされるべきコードおよび／またはデータの実行可能な画像を特定することを含み得る。このコードは、それがエンクレーブコンテナ内へロードされる前または後に、インスタンス化されたエンクレーブのアテステーションのために使用される（図５に関して上に説明されるように）初期状態の部分になり得る。例えば、エンクレーブの実行可能な画像（エンクレーブバイナリ）は、実行可能な画像を含むメモリ内のバッファにポインターを提供することによって、エンクレーブクライアントにより特定され得る。代替的に、エンクレーブ画像は、エンクレーブバイナリを含むファイルシステム内のファイルを示すことによって特定され得る。いくつかの実施形態において、特定されたエンクレーブ画像は、暗号化され得、他の実施形態において、エンクレーブは、暗号化されない場合があり、他の実施形態において、エンクレーブは、部分的に暗号化され得る。アテステーションのためのエンクレーブバイナリの測定は、暗号化された実行可能な画像に対して、または復号の後に発生し得る。

【００５６】

[0078]エンクレーブ内へ最初にロードされるべきコードおよび／またはデータは、エンクレーブ一次画像を含むファイルを特定することによって示され得る。このコードおよび／またはデータに加えて、エンクレーブ一次画像は、エンクレーブの所望のサイズ（エンクレーブコンテナの内側で必要とされるメモリの量）、ファイル内のコード内でのエントリーポイントの場所、および依存画像ファイルのリストなどの追加のメタデータを含み得る。依存画像ファイルは、一次画像ファイル内のコードおよびデータと一緒にエンクレーブ内へ同様にロードされ得る他の（非一次）画像ファイルである。依存画像ファイルは、それら自体がさらなる依存画像ファイルのリストを含み得る。図９のローカルエンクレーブシステムの場合、一次画像および依存画像は、ローカルでアクセス可能なファイルシステムを介するなど、任意のアクセス可能な記憶デバイス内にファイルされ得る。図１０のリモートエンクレーブシステムの場合、一次画像ファイルは、コンピューター１０１０またはコンピューター１０５０のいずれかにアクセス可能な任意の記憶デバイス内にあり得

る。クライアント１０５６が、コンピューター１０５０上に位置する一次画像を使用してコンピューター１０１０上でのエンクレーブの作成をリクエストする場合、抽象化リモート処理プラットフォームおよびクライアントスタブ１０１６は、特定された一次画像ファイルをコンピューター１０１０にコピーするために連携し得る。

【００５７】

[0079]ＣｒｅａｔｅＥｎｃｌａｖｅは、エンクレーブをインスタンス化するための例示的な方法である。ＣｒｅａｔｅＥｎｃｌａｖｅ方法は、疑似コードを用いて説明され得る。

```
ＨＡＮＤＬＥ
ＣｒｅａｔｅＥｎｃｌａｖｅ（
_Ｉｎ_　ＰＣＷＳＴＲ　ｅｎｃｌａｖｅＰａｔｈ，
_Ｉｎ_　ＤＷＯＲＤ　ｆｌＥｎｃｌａｖｅＴｙｐｅ，
_Ｉｎ_　ＤＷＯＲＤ　ｄｗＦｌａｇｓ，
_Ｉｎ_ｒｅａｄｓ_ｂｙｔｅｓ_（ｄｗＩｎｆｏＬｅｎｇｔｈ）
ＰＣＶＯＩＤ　ｅｎｃｌａｖｅＩｎｆｏｒｍａｔｉｏｎ，
_Ｉｎ_　ＤＷＯＲＤ　ｄｗＩｎｆｏＬｅｎｇｔｈ，
_Ｏｕｔ_ｏｐｔ_　ＰＤＷＯＲＤ　ｅｎｃｌａｖｅＥｒｒｏｒ
）
```

[0080]本明細書内の方法を説明するために使用される疑似コードは、ＡＰＩインターフェースを規定するためのいくつかの疑似コード変換を使用し得る。例えば、上のｅｎｃｌａｖｅＰａｔｈなどの関数パラメーターは、パラメーターが入力または出力パラメーターであることをそれぞれ示すために「_Ｉｎ_」または「_Ｏｕｔ_」が施され得る。「_Ｏｕｔ_ｏｐｔ_」は、任意選択の出力パラメーターを示し得る。すべて大文字の用語は、データタイプを示し得る。ＨＡＮＤＬＥは、３２ビットの数字など、何かを間接的に指すために使用される数字であり得る。例えば、上のＣｒｅａｔｅＥｎｃｌａｖｅ方法は、ＣｒｅａｔｅＥｎｃｌａｖｅの呼び出し元にＨＡＮＤＬＥを返し、そのＨＡＮＤＬＥは、作成されたエンクレーブのハンドルであり得、ＰＣＷＳＴＲは、特定のタイプの文字列へのポインターであり得、ＤＷＯＲＤは、未署名の３２ビット量であり得、ＰＣＶＯＩＤは、未特定のタイプのデータへのポインターであり得、ＢＯＯＬは、バイナリ値であり得る。

【００５８】

[0081]ＣｒｅａｔｅＥｎｃｌａｖｅは、クライアント９１６などのクライアントが、エンクレーブを作成し、エンクレーブ内で一次画像をロードすることを可能にし得る。この画像内の任意のエンクレーブ構成情報は、インスタンス化されたエンクレーブと関連付けられ得る。ＣｒｅａｔｅＥｎｃｌａｖｅは、以下のパラメーターを含み得る。

【００５９】

ｌｐＥｏｃｌａｖｅＮａｍｅ：エンクレーブ一次画像へのパスを特定し得、これは、実装形態において、オープンされたファイルへのハンドル、Ｕｎｉｆｏｒｍ　Ｒｅｓｏｕｒｃｅ　Ｉｄｅｎｔｉｆｉｅｒ（ＵＲＩ）、または外部ルックアップ内で使用される識別子など、エンクレーブ一次画像のコードおよび／またはデータを識別するための何らかの他のタイプの識別子であり得る。例えば、グローバル一意識別子（ＧＵＴＤ）が、一次画像のデータベース内への鍵として使用され得る。他の実装形態において、このパラメーターは、エンクレーブ一次画像を含むメモリ領域を識別し得る。

【００６０】

ｆｌＥｎｃｌａｖｅＴｙｐｅ：作成するべきエンクレーブのタイプを特定し得る（エンクレーブ画像が複数のタイプをサポートする場合）。バイナリが１つのエンクレーブのみをサポートするか、または開発者がデフォルトを明示的に特定している場合にはＤＥＦＡＵＬＴに設定され得る。ｄｗＦｌａｇｓ：エンクレーブを作成してエンクレーブ一次画像をロードするときにとられるべき１つまたは複数の既定の行為を特定し得る。

【００６１】

ｅｎｃｌａｖｅＩｎｆｏｒｍａｔｉｏｎ：エンクレーブのランタイム構成のための任意選択の入力パラメーターであり得る。

ｌｐＥｏｃｌａｖｅＥｒｒｏｒ：アーキテクチャ特有のエラーコードを返すために任意選択のパラメーターを特定し得る。

【００６２】

[0082]正常に完了すると、ＣｒｅａｔｅＥｎｃｌａｖｅは、エンクレーブにハンドルを返し得る。エラー時には、ＮＵＬＬが返され得る。他の識別子（ＧＵＩＤ、ＵＲＩなど）も、本開示の範囲から逸脱することなく返され得る。簡略性のため、本明細書は、ハンドルを使用してＡＰＩを説明するものとする。エンクレーブ作成は、例えば、エンクレーブメモリの欠如、抽象化プラットフォームもしくはネイティブプラットフォーム内の特定されたエンクレーブタイプに対するサポートの欠如に起因して失敗し得るか、または、作成は、特定されたタイプのエンクレーブがシステム上で実行することを防ぐ明示的な構成ポリシーに起因して失敗し得る。

【００６３】

[0083]ＣｒｅａｔｅＥｎｃｌａｖｅおよび以下に説明される他のＡＰＩ方法の実装形態は、説明される方法パラメーターのうちの１つまたは複数を除外し得る。例えば、ＣｒｅａｔｅＥｎｃｌａｖｅに関して、ｌｐＥｎｃｌａｖｅＮａｍｅ、ｆｌＥｎｃｌａｖｅＴｙｐｅ、ｄｗＦｌａｇｓ、およびｅｎｃｌａｖｅＩｎｆｏｒｍａｔｉｏｎは、その特定のＡＰＩのための特定の既定の値を使用して、除外され得る。ｌｐＥｎｃｌａｖｅＥｒｒｏｒ論もまた、ＡＰＩから除外され得、ＡＰＩ呼び出し内のエラーをチェックするための代替的な方法が、任意選択的に実施され得る。

【００６４】

[0084]ＣｒｅａｔｅＥｎｃｌａｖｅは、エンクレーブ一次画像内で特定されるようなすべての依存モジュールをロードすることを担い得る。エンクレーブ一次画像は、一次画像が依存する他のバイナリ画像ファイルを特定するポータブル実行（ＰＥ）ファイルであり得る。ＣｒｅａｔｅＥｎｃｌａｖｅはまた、アテステーションのための測定を仕上げること、トランスポート層セキュリティ（ＴＬＳ）ならびに／または他の鍵合意および通信プロトコルのための構造を割り当てることなど、ネイティブプラットフォーム特有の初期化を実施し得る。エンクレーブ抽象化プロトコルインターフェース９２０、９２２（例えば、データ密封およびアテステーションのための方法を含む）は、エンクレーブ初期化が完了した後に動作可能であり得る。

【００６５】

[0085]ＴｅｒｍｉｎａｔｅＥｎｃｌａｖｅは、エンクレーブを終了するための例示的な方法である。

```
ＶＯＩＤ
ＴｅｒｍｉｎａｔｅＥｎｃｌａｖｅ（
＿Ｉｎ＿　ＨＡＮＤＬＥ　ｈＥｎｃｌａｖｅ
）
```

【００６６】

[0086]ＴｅｒｍｉｎａｔｅＥｎｃｌａｖｅは、エンクレーブを破壊するために使用され得る。実装形態において、エンクレーブを破壊することは、強制的にすべてのエンクレーブスレッドをホストに返すかもしくは終了すること、および／またはエンクレーブと関連付けられたメモリを解放することを含み得る。実行中のエンクレーブ上でＴｅｒｍｉｎａｔｅＥｎｃｌａｖｅを呼び出すことは、実行中のエンクレーブを終了して、エンクレーブと関連付けられたすべてのリソースを解放し得る。

【００６７】

[0087]エンクレーブ抽象化プラットフォーム９１２は、例えば、エンクレーブとそのクライアントとの間で制御を転送するために使用され得る実行制御転送プリミティブを含み得る。実行制御転送プリミティブは、他のコンポーネント内のエントリーポイントにおいてコードの実行を開始することによってエンクレーブ９１４とクライアント９１６との間

の通信を可能にし得る。実行制御転送プリミティブは、パラメーターが制御転送リクエストと関連付けられることを可能にすることによってデータをエンクレーブ内へ／から外へ引き渡すことを可能にし、パラメーターは、個々のデータアイテムを特定し得る（パラメーター自体が通信される）か、またはパラメーターは、メモリ領域へのポインターであり得る（パラメーターによってポイントされるバッファが通信される）。これらのプリミティブは、エンクレーブコンテナに対するセキュリティ制限に起因するエンクレーブ９１４とクライアント９１６との間の直接的な呼び出しまたは飛び越しへの制約にもかかわらず、制御転送を可能にし得る。

【００６８】

[0088]エンクレーブへのコールインの場合、インターフェース９２４は、クライアント９１６がインターフェース９２０を介してエンクレーブ９１４へコールインすることを可能にする機序を含み得る。例えば、インターフェース９２４は、ＧｅｔＰｒｏｃＡｄｄｒｅｓｓおよびＣａｌｌＥｎｃｌａｖｅ方法を含み得る。

```
ｔｙｐｅｄｅｆ　ＰＶＯＩＤ（＊ＥＮＣＰＲＯＣ）（ＰＶＯＩＤ）；
ＥＮＣＰＲＯＣ
ＧｅｔＰｒｏｃＡｄｄｒｅｓｓ（
＿Ｉｎ＿　ＨＭＯＤＵＬＥ　ｈＥｎｃｌａｖｅ，
＿Ｉｎ＿　ＬＰＣＴＳＴＲ　ｌｐＰｒｏｃＮａｍｅ
）
ＢＯＯＬ
ＣａｌｌＥｎｃｌａｖｅＩｎ（
＿Ｉｎ＿　ＥＮＣＰＲＯＣ　ｐＣａｌｌｉｎ，
＿Ｉｎ＿　ＰＶＯＩＤ　ｐＰａｒａｍｅｔｅｒ，
＿Ｏｕｔ＿　ＰＶＯＩＤ　ｐＲｅｔｕｒｎ
）
```

【００６９】

[0089]クライアント９１６などのエンクレーブクライアントは、ＧｅｔＰｒｏｃＡｄｄｒｅｓｓ（）によって返される関数ポインターを使用して、エンクレーブ９１４などのエンクレーブへコールインすることができる。ｌｐＰｒｏｃＮａｍｅパラメーターは、エンクレーブ一次画像内にエクスポートされる関数と一致し得る。例えば、

```
／／エンクレーブのためのＣａｌｌｉｎ関数を呼び出す。
ＥＮＣＰＲＯＣ　ｐＣａｌｌｉｎ　＝　（ＥＮＣＰＲＯＣ）　ＧｅｔＰｒｏｃＡｄｄｒｅｓｓ（ｈＥｎｃｌａｖｅ，”ＣａｌｌｉｎＥｘａｍｐｌｅ”）；
ＰＶＯＩＤ　ｐＰａｒａｍｅｔｅｒ；／／メモリへのポインター
ｉｆ（ＮＵＬＬ　！＝ｐＣａｌｌｉｎ）
{
ＣａｌｌＥｎｃｌａｖｅＩｎ（ｐＣａｌｌｉｎ，ｐＰａｒａｍｅｔｅｒ）；
}
```

【００７０】

[0090]ＧｅｔＰｒｏｃＡｄｄｒｅｓｓの他の実施形態において、ｌｐＰｒｏｃＮａｍｅは、エンクレーブ画像からエクスポートされるエントリーポイントの列挙からの選択など、数字などの特定のエクスポートされた関数の別の識別子、または関数に対応する他の非テキスト識別子であり得る。ＣａｌｌＥｎｃｌａｖｅＩｎの他の実施形態は、追加的に、コールインされるべきエンクレーブを特定する入力パラメーター、例えば、ハンドルを返されたＣｒｅａｔｅＥｎｃｌａｖｅをとり得る。

【００７１】

[0091]エンクレーブへコールインするとき、クライアントプロセス内のスレッドは、保留され得、エンクレーブスレッド（別個のスレッドＩＤ付き）は、コールインリクエストをサービスするために使用され得る。エンクレーブスレッド上で実行するエンクレーブコードは、このとき、エンクレーブへのコールインの前に以前はエンクレーブクライアント

に利用可能であったメモリへのアクセスを有し得る。例えば、クライアントは、ＣａｌｌＥｎｃｌａｖｅＩｎ抽象化方法を呼び出す前にｐＰａｒａｍｅｔｅｒによってポイントされるバッファ内へデータを入れ得、その後エンクレーブは、コールインリクエストをサービスしながら、ｐＰａｒａｍｅｔｅｒによってポイントされるバッファへのアクセスを有し得る。コールアウトの際、元の（クライアント）呼び出しスレッドが、コールアウトをサービスするために使用され得る。リエントラント性がサポートされ得る（例えば、ホスト内のコールアウトは、再びエンクレーブへコールインすることができる）。

【００７２】

[0092]エンクレーブのコールアウトの場合、インターフェース９２２は、エンクレーブ９１４がエンクレーブクライアント９１６へコールアウトすることを可能にする上のＣａｌｌＥｎｃｌａｖｅＩｎ方法に関連した方法を含み得る。例えば、エンクレーブ９１４は、特定のタイプ、例えばＥＮＣＰＲＯＣ関数タイプ、のホストプロセス内で任意の関数をコールアウトし得る。ＥＮＣＰＲＯＣ関数タイプの関数ポインターは、エンクレーブへのコールインパラメーターを使用して引き渡され得る。

```
ＢＯＯＬ
ＣａｌｌＥｎｃｌａｖｅＯｕｔ（
＿Ｉｎ＿　ＥＮＣＰＲＯＣ　ｐＣａｌｌｏｕｔ，
＿Ｉｎ＿　ＰＶＯＩＤ　ｐＰａｒａｍｅｔｅｒ，
＿Ｏｕｔ＿　ＰＶＯＩＤ　ｐＲｅｔｕｒｎ
）
／／ホストプロセス内の関数へコールアウトする
ＥＮＣＰＲＯＣ　ｐＣａｌｌｏｕｔ＝（ＥＮＣＰＲＯＣ）０ｘＦ００；／／ホスト内の何らかの関数へのアドレス
ＰＶＯＩＤ　ｐＰａｒａｍｅｔｅｒ＝／／メモリへのポインター
ＣａｌｌＥｎｃｌａｖｅＯｕｔ（ｐＣａｌｌｏｕｔ，ｐＳｈａｒｅｄＭｅｍｏｒｙ）；
```

インターフェース９２０は、上の「ＣａｌｌｉｎＥｘａｍｐｌｅ」関数として登録されたエントリーポイントを含み得、インターフェース９２６は、上の「Ｃａｌｌｏｕｔ」関数として登録されたエントリーポイントを含み得る。例えば、エンクレーブ一次画像がポータブル実行可能な（ＰＥ）画像形式にある場合、画像内の関数エントリーポイントは、「エクスポート」エントリーポイントとしてリストされ得、各そのようなエクスポートされたエントリーポイントは、そのエンクレーブＰＥ画像内のエントリーポイントを識別および区別するために「ＣａｌｌｉｎＥｘａｍｐｌｅ」などのテキスト名を含み得、他の実装形態において、関数エントリーポイントは、関数がエンクレーブのためのエントリーポイントであり得ることを示す１ビットなど、追加のメタデータでマークされ得る。エンクレーブのコールアウトのための上の例において、コールアウト関数のアドレスは、０ｘＦ００として得られ、単に例にすぎない。コールアウト関数の実際のアドレスは、様々なやり方で決定され得る。例えば、クライアントの内側のコールアウト関数アドレスは、コールイン関数のためのパラメーターとしてエンクレーブ内へ引き渡され得る。別の例において、コールアウト関数のアドレスは、ＲｅｇｉｓｔｅｒＣａｌｌＯｕｔなどの関数を使用してクライアントによって登録され得る。

```
ＢＯＯＬ　ＲｅｇｉｓｔｅｒＣａｌｌＯｕｔ（
＿Ｉｎ＿　ＥＮＣＰＲＯＣ　ｐＣａｌｌｏｕｔ，
＿Ｉｎ＿　ＬＰＣＴＳＴＲ　ｌｐＰｒｏｃＮａｍｅ）
```

エンクレーブの内側のコードは、ＧｅｔＣａｌｌＯｕｔなどの補関数を呼び出すことによってコールアウト関数のアドレスを獲得し得る。

```
ＢＯＯＬ　ＧｅｔＣａｌｌＯｕｔ（
＿Ｏｕｔ＿　ＥＮＣＰＲＯＣ　＊ｐＣａｌｌｏｕｔ，
＿Ｉｎ＿　ＬＰＣＴＳＴＲ　ｌｐＰｒｏｃＮａｍｅ）
```

[0093]他の実施形態において、ＣａｌｌＥｎｃｌａｖｅＩｎおよびＣａｌｌＥｎｃｌａｖｅＯｕｔ方法は、実際には同じ方法であり得る。例えば、単一のＣａｌｌＥｎｃｌａｖ

ｅ方法が、エンクレーブへコールインするためおよびエンクレーブからコールアウトするために使用され得る。エンクレーブクライアント９１６もまたエンクレーブである状況において、クライアント９１６へのエンクレーブ９１４のコールアウトは、エンクレーブへのコールインでもある。

【００７３】

[0094]抽象化プラットフォーム９１２は、データをエンクレーブへ密封するためのプリミティブを提供し得る。例えば、インターフェース９２２は、データをエンクレーブアイデンティティへ密封および開封することなど、エンクレーブ９１４へのサービスを提供し得る。上に説明されるように、エンクレーブは、複数のネスト化アイデンティティを有し得、データは、任意のそのようなアイデンティティへと密封され得る。データが、潜在的なエンクレーブインスタンス化のセットに対応するアイデンティティへ密封されるとき、密封データは、エンクレーブインスタンス化のその対応するセットのいずれかによって開封され得る。例えば、

```
ｓｔｒｕｃｔ　ＳＥＡＬＩＮＧ　ＰＯＬＩＣＹ
{
ＥＮＣＬＡＶＥ＿ＩＤ＿ＴＹＰＥ　ｅｎｃｌａｖｅＩｄＴｙｐｅ；
}；
```

【００７４】

[0095]ｅｎｃｌａｖｅＩｄＴｙｐｅの各値については、エンクレーブは、マッピングＩＤへと密封する。潜在的なエンクレーブアイデンティティタイプ（およびｅｎｃｌａｖｅＩｄＴｙｐｅの値）としては、以下が挙げられる。

```
ＥＮＣＬＡＶＥ＿ＥＸＡＣＴＨＡＳＨ
ＥＮＣＬＡＶＥ＿ＩＮＳＴＡＮＣＥＨＡＳＨ：　／／ＳＧＸの場合はＭＲＥＮＣＬＡＶＥ、ＶＳＭの場合はＩＭＡＧＥ　ＨＡＳＨを使用した密封
ＥＮＣＬＡＶＥ＿ＩＭＡＧＥＩＤＳ：　／／ＳＧＸではサポートされず、ＶＳＭの場合ＩＭＡＧＥ　ＩＤＳを使用する
ＥＮＣＬＡＶＥ＿ＦＡＭＩＬＹＩＤ：　／／ＳＧＸの場合はＰＲＯＤＵＣＴＩＤ、ＶＳＭの場合はＦＡＭＩＬＹ　ＩＤを使用する
ＥＮＣＬＡＶＥ＿ＡＵＴＨＯＲＩＤ：　／／ＳＧＸの場合はＭＲＳＩＧＮＥＲ、ＶＳＭの場合はＡＵＴＨＯＲ　ＩＤを使用する
```

【００７５】

[0096]プラットフォームはまた、追加のデバッグ構成（著作されたおよびランタイム）を密封ポリシーに適用し得る。異なるデバッグポリシーでは、異なる密封鍵が使用され得る。例えば、デバッグおよびリリース構成は、異なる密封鍵を使用し得る。

```
ＤＷＯＲＤ
Ｅｎｃｌａｖｅ　Ｓｅａｌ（
＿Ｉｎ＿　ＳＥＡＬＩＮＧ＿ＰＯＬＩＣＹ　ｓｅａｌｉｎｇＰｏｌｉｃｙ，
＿Ｉｎ＿ｒｅａｄｓ＿ｂｙｔｅｓ＿ｏｐｔ＿（ｄｗＰｌａｉｎｔｅｘｔＳｉｚｅ）ＬＰＣＶＯＩＤ　ｐＰｌａｉｎｔｅｘｔ，
＿Ｉｎ＿　ＤＷＯＲＤ　ｄｗＰｌａｉｎｔｅｘｔＳｉｚｅ，
＿Ｉｎ＿ｒｅａｄｓ＿ｂｙｔｅｓ＿ｏｐｔ＿（ｄｗＡｕｔｈｄａｔａＳｉｚｅ）ＬＰＣＶＯＩＤ　ｐＡｕｔｈｄａｔａ，
＿Ｉｎ＿　ＤＷＯＲＤ　ｄｗＡｕｔｈｄａｔａＳｉｚｅ，
＿Ｏｕｔ＿ｗｒｉｔｅｓ＿ｂｙｔｅｓ＿ｔｏ＿（ｄｗＳｅａｌｅｄｔｅｘｔＳｉｚｅ）ＬＰＶＯＩＤ　ｐＳｅａｌｅｄｔｅｘｔ，
＿Ｉｎｏｕｔ＿　ＤＷＯＲＤ　＊ｄｗＳｅａｌｅｄｔｅｘｔＳｉｚｅ
）
ＤＷＯＲＤ
ＥｎｃｌａｖｅＵｎｓｅａｌ（
＿Ｉｎ＿ｒｅａｄｓ＿ｂｙｔｅｓ＿ｏｐｔ＿（ｄｗＳｅａｌｅｄｔｅｘｔＳｉｚｅ）ＬＰ
```

```
CVOID pSealedtext,
_In_ DWORD dwSealedtextSize,
_In_reads_bytes_opt_(dwAuthdataSize) LPCV
OID pAuthdata,
_In_ DWORD dwAuthdataSize,
_Out_writes_bytes_to_(dwPlaintextSize) LP
CVOID pPlaintext,
_Inout_ DWORD *dwPlaintextSize
)
```

【００７６】

[0097]抽象化プラットフォーム９１２は、アテステーションレポートおよびクォートをもたらすため、ならびにレポートおよびクォートを検証するためなど、アテステーションのためのプリミティブを提供し得る。例えば、

```
DWORD
CreateReport(
_In_reads_bytes_opt_(dwTargetInfoSize) PC
VOID pTargetlnfo,
_In_ DWORD dwTargetInfoSize,
_In_reads_bytes_opt_(dwAuthData) PCVOID p
AuthData,
_In_ DWORD dwAuthData,
_Out_writes_bytes_opt_(*pReportSize) PVOI
D pReport,
_Inout_ PDWORD pReportSize,
_Out_opt_ PDWORD IpEnclaveError
)
DWORD
VerifyReport(
_In_reads_bytes_(dwReportSize) PCVOID pRe
port,
_In_ DWORD dwReportSize,
_Out_opt_ LPDWORD lpEnclaveError
)
```

【００７７】

[0098]VerifyReport()は、レポートの完全性、およびレポートが同じマシン上のエンクレーブによって生成されたことを確認するためにエンクレーブによって使用され得る。

```
DWORD CreateQuote(
_In_ GUID quoteType,
_In_ DWORD authDataSize,
_In_reads_bytes_opt_(authDataSize) const
BYTE* authData,
_Out_ DWORD* quoteSize,
_Outptr_result_bytebuffer_opt_(*quoteSiz
e) BYTE** quote
)
```

【００７８】

[0099]CreateQuoteにおいて、quoteTypeは、特定のクォートを生成するための信頼の源であり得るクォートプロバイダーへマッピングし得る。CreateQuoteにおいて、authDataは、CreateQuoteの呼び出し元によ

って作成される、およびＣｒｅａｔｅＱｕｏｔｅの呼び出し元によって規定される形式にあるデータへのポインターであり得る。ａｕｔｈＤａｔａは、抽象化プラットフォーム９１２によって理解される必要がないことに留意されたい。ａｕｔｈＤａｔａは、結果として生じるクォート内へ詰め込まれ得る。クォートプロバイダーは、これをサポートすることが期待され得る。

```
DWORD VerifyQuote(
_In_ DWORD quoteSize,
_In_reads_bytes_(quoteSize) const BYTE* quote,
_Out_ DWORD* reportSize,
_Outptr_result_bytebuffer_all_(*reportSize) BYTE** report
)
```

【００７９】

[0100]上に説明されるエンクレーブプリミティブに加えて、エンクレーブ抽象化プラットフォームは、メモリ管理（例えば、エンクレーブに限定されているメモリ、エンクレーブとそのクライアントとの間で共有されるメモリなどのメモリを割り当てる、および解放するため）、例外処理（例えば、エンクレーブコードを実行している間に発生するエラーまたは例外を処理するため）、スレッド同期、および暗号関数（例えば、暗号化、ハッシュ関数、および署名）を提供し得る。

【００８０】

[0101]上に説明される技術は、以下に説明されるように、１つまたは複数のコンピューティングデバイスまたは環境上で実施され得る。図１１は、本明細書に説明される技術の一部が具現化され得る、例えば、信頼できるハードウェア１７２、トラステッドプラットフォーム７３６、またはコンピューター８１０、９１０、１０１０、および１０５０のうちの１つまたは複数を具現化し得る例示的な汎用コンピューティング環境を描写する。コンピューティングシステム環境１１０２は、好適なコンピューティング環境の一例にすぎず、本開示される主題の用途または機能性の範囲に関していかなる限定も提示することは意図されない。コンピューティング環境１１０２は、例示的な動作環境１１０２内に例証されるコンポーネントの任意の１つまたは組合せに関して何らかの依存関係または要件を有すると解釈されないものとする。いくつかの実施形態において、様々な描写されたコンピューティング要素は、本開示の特定の態様を例示化するように構成される回路を含み得る。例えば、本開示で使用される回路という用語は、ファームウェアまたはスイッチによって関数を実施するように構成される特殊ハードウェアコンポーネントを含み得る。他の例示的な実施形態において、回路という用語は、関数を実施するように動作可能な論理を具現化するソフトウェア命令によって構成される汎用処理ユニット、メモリなどを含み得る。回路がハードウェアおよびソフトウェアの組合せを含む例示的な実施形態において、実装者は、論理を具現化するソースコードを書くことができ、ソースコードは、汎用処理ユニットによって処理され得る機械可読コードにコンパイルされ得る。当業者は、先行技術が、ハードウェア、ソフトウェア、またはハードウェア／ソフトウェアの組合せの間にはほんの少しの差しかないところまで発展したということを理解し得るため、特定の関数を達成するためのハードウェア対ソフトウェアの選択は、実装者に委ねられたデザイン選択である。より詳細には、当業者は、ソフトウェアプロセスが等価のハードウェア構造に転換され得、ハードウェア構造は、それ自体が等価のソフトウェアプロセスに転換され得るということを理解し得る。したがって、ハードウェア実装形態対ソフトウェア実装形態の選択は、実装者に委ねられたデザイン選択のうちの１つである。

【００８１】

[0102]モバイルデバイスもしくはスマートフォン、タブレット、ラップトップ、デスクトップコンピューター、またはネットワーク化デバイス、クラウドコンピューティングリソースなどの集合のいずれかを含み得るコンピューター１１０２は、典型的には、様々な

コンピューター可読媒体を含む。コンピューター可読媒体は、コンピューター１１０２によってアクセスされ得、かつ揮発性および不揮発性媒体、可換型および非可換型媒体の両方を含む、任意の利用可能な媒体であり得る。システムメモリ１１２２は、リードオンリメモリ（ＲＯＭ）１１２３およびランダムアクセスメモリ（ＲＡＭ）１１６０などの揮発性および／または不揮発性メモリの形態でコンピューター可読記憶媒体を含む。起動中などに、コンピューター１１０２内の要素間で情報を転送するのに役立つ基本ルーチンを含む基本入力／出力システム１１２４（ＢＩＯＳ）は、典型的には、ＲＯＭ１１２３に格納される。ＲＡＭ１１６０は、典型的には、直ちにアクセス可能である、および／または処理ユニット１１５９上で現在動作されているデータおよび／またはプログラムモジュールを含む。例として、また限定ではなく、図１１は、ハイパーバイザー１１３０、オペレーティングシステム（ＯＳ）１１２５、アプリケーションプログラム１１２６、エンクレーブクライアント１１６５を含む他のプログラムモジュール１１２７、およびエンクレーブ１１２８を例証する。

【００８２】

[0103]コンピューター１１０２はまた、他の可換型／非可換型の揮発性／不揮発性コンピューター記憶媒体を含み得る。単に例として、図１１は、非可換型の不揮発性磁気媒体から読み込むまたはそこへ書き込むハードディスクドライブ１１３８、可換型の不揮発性磁気ディスク１１５４から読み込むまたはそこへ書き込む磁気ディスクドライブ１１３９、およびＣＤ　ＲＯＭまたは他の光学媒体などの可換型の不揮発性光学ディスク１１５３から読み込むまたはそこへ書き込む光学ディスクドライブ１１０４を例証する。例示的な動作環境において使用され得る他の可換型／非可換型の揮発性／不揮発性コンピューター記憶媒体としては、限定されるものではないが、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、固体ＲＡＭ、固体ＲＯＭ、および同様のものが挙げられる。ハードディスクドライブ１１３８は、典型的には、インターフェース１１３４などの非可換型メモリインターフェースを通じてシステムバス１１２１に接続され、磁気ディスクドライブ１１３９および光学ディスクドライブ１１０４は、典型的には、インターフェース１１３５または１１３６などの可換型メモリインターフェースによってシステムバス１１２１に接続される。

【００８３】

[0104]上で論じられかつ図１１に例証されるドライブおよびそれらの関連コンピューター記憶媒体は、コンピューター可読命令、データ構造、プログラムモジュール、およびコンピューター１１０２の他のデータの記憶装置を提供する。図１１では、例えば、ハードディスクドライブ１１３８は、オペレーティングシステム１１５８、アプリケーションプログラム１１５７、エンクレーブクライアントアプリケーションおよびエンクレーブバイナリファイルなどの他のプログラムモジュール１１５６、ならびにプログラムデータ１１５５を格納するものとして例証される。これらのコンポーネントは、オペレーティングシステム１１２５、アプリケーションプログラム１１２６、他のプログラムモジュール１１２７、およびプログラムデータ１１２８と同じであるか、またはそれらと異なるかのいずれかであり得ることに留意されたい。オペレーティングシステム１１５８、アプリケーションプログラム１１５７、他のプログラムモジュール１１５６、およびプログラムデータ１１５５は、少なくともそれらが異なるコピーであることを例証するためにここでは異なる数字が付与される。ユーザーは、一般にはマウス、トラックボール、またはタッチパッドと称されるキーボード１１５１およびポインティングデバイス１１５２などの入力デバイスを通じて、コマンドおよび情報をコンピューター１１０２へ入力し得る。他の入力デバイス（図示されない）は、マイク、ジョイスティック、ゲームパッド、パラボラアンテナ、スキャナー、網膜スキャナー、または同様のものを含み得る。これらおよび他の入力デバイスは、多くの場合、システムバス１１２１に結合されるユーザー入力インターフェース１１３６を通じて処理ユニット１１５９に接続されるが、パラレルポート、ゲームポート、またはユニバーサルシリアルバス（ＵＳＢ）などの他のインターフェースおよびバス構造によって接続されてもよい。モニター１１４２または他のタイプのディスプレイデ

バイスもまた、ビデオインターフェース１１３２などのインターフェースを介してシステムバス１１２１に接続される。モニターに加えて、コンピューターはまた、出力周辺インターフェース１１３３を通じて接続され得るスピーカー１１４４およびプリンター１１４３などの他の周辺出力デバイスを含み得る。

【００８４】

[0105]コンピューター１１０２は、リモートコンピューター１１４６などの１つまたは複数のリモートコンピューターへの論理接続を使用してネットワーク化環境内で動作し得る。リモートコンピューター１１４６は、パーソナルコンピューター、サーバー、ルーター、ネットワークＰＣ、ピアデバイス、または他の共通ネットワークノードであり得、典型的には、コンピューター１１０２に関連して上に説明される要素の多くまたはすべてを含むが、メモリ記憶デバイス１１４７のみが図１１には例証されている。図１１に描写される論理接続は、ローカルエリアネットワーク（ＬＡＮ）１１４５および広域ネットワーク（ＷＡＮ）１１４９を含むが、他のネットワークも含み得る。そのようなネットワーキング環境は、オフィス、企業全体のコンピューターネットワーク、イントラネット、インターネット、およびクラウドコンピューティングリソースでは一般的である。

【００８５】

[0106]ＬＡＮネットワークキング環境が使用されるとき、コンピューター１１０２は、ネットワークインターフェースまたはアダプター１１３７を通じてＬＡＮ１１４５に接続される。ＷＡＮネットワーキング環境が使用されるとき、コンピューター１１０２は、典型的には、モデム１１０５、または、インターネットなどの、ＷＡＮ１１４９を介した通信を確立するための他の手段を含む。内部または外部であり得るモデム１１０５は、ユーザー入力インターフェース１１３６または他の適切な機序を介してシステムバス１１２１に接続され得る。ネットワーク化環境において、コンピューター１１０２に関して描写されるプログラムモジュールまたはその部分は、リモートメモリ記憶デバイス内に格納され得る。例として、また限定ではなく、図１１は、リモートアプリケーションプログラム１１４８をメモリデバイス１１４７上に存在するものとして例証する。示されるネットワーク接続は例であり、コンピューター同士の通信リンクを確立する他の手段が使用され得ることを理解されたい。

【００８６】

[0107]図１２は、ネイティブエンクレーブプラットフォームを抽象化する方法１２００の例示的なフローチャートを描写する。図９の９１２などの抽象化プラットフォームは、ボックス１２０２においてエンクレーブプラットフォームのためのリクエストを受信し得る。リクエストは、クライアント９１４などのエンクレーブクライアント、またはエンクレーブ９１６などのエンクレーブからのものであり得る。

【００８７】

[0108]エンクレーブからのリクエストは、抽象化プラットフォームプリミティブを実施するためのリクエストであり得、例えば、エンクレーブのアテステーションレポートまたはクォートを作成するためのリクエスト、エンクレーブへとデータを密封するためのリクエスト、エンクレーブのクライアント内に関数を呼び出す（クライアントへコールアウトする）ためのリクエスト、単調カウンターを読み込む（単調カウンターの現在の値を提供する）ためのリクエスト、信頼できる時刻測定値を提供するためのリクエスト、およびエンクレーブとそのクライアントとの間で共有され得るメモリを割り当てるためのリクエスト（例えば、エンクレーブへのコールインまたはエンクレーブからのコールアウトの際に、共有メモリへのポインターがパラメーターとして引き渡されることを可能にするため）を含み得る。いくつかの実施形態において、エンクレーブクライアントの仮想メモリ空間全体は、共有メモリを割り当てるためのリクエストがメモリをエンクレーブのクライアントに割り当てるためのリクエストとして実施され得るように、エンクレーブと共有され得る（およびそこからアクセス可能である）。いくつかの実施形態において、共有メモリを割り当てる方法は、エンクレーブおよびそのクライアントの両方に利用可能である。

【００８８】

[0109]エンクレーブクライアントからのリクエストは、抽象化プラットフォームプリミティブを実施するためのリクエストであり得、例えば、エンクレーブをインスタンス化するためのリクエスト、エンクレーブのアテステーションレポートを検証するためのリクエスト、エンクレーブの内側で関数を呼び出す（エンクレーブへコールインする）ためのリクエスト、およびエンクレーブとそのクライアントとの間で共有され得るメモリを割り当てるためのリクエストを含み得る。

【００８９】

[0110]抽象化プラットフォームリクエストは、動作１２０４～１２０８においてネイティブプラットフォームリクエストへと翻訳され得る。受信したリクエストに含まれるか、またはその中で暗示されるパラメーターは、例えば、元のリクエスト内のパラメーターのデータ形式がネイティブプラットフォーム内のそのパラメーターのデータ形式と同じではないことが決定される場合、任意選択のステップ１２０４において変換され得る。例えば、エンクレーブまたはクライアントからのリクエストが、エンクレーブ抽象化アイデンティティなどの抽象化形式アテステーションレポートから導出されるパラメーターを含む場合、ネイティブエンクレーブアイデンティティなどのネイティブ形式アテステーションレポート内で使用されるパラメーターに変換されることになる。

【００９０】

[0111]ネイティブプラットフォームおよび受信したリクエストの呼び出し規則が異なることが決定される場合、呼び出し規則は、任意選択のステップ１２０６において変換され得る。呼び出し規則は、例えば、呼び出しスタック上のパラメーターを並べ替えること、プロセッサーレジスターと呼び出しスタックとの間でパラメーターを移動させること、ならびに、エラー値を返すことおよび例外ハンドラーを呼び出すことなどのエラー状態通信方法間で変換することによって変換され得る。

【００９１】

[0112]いくつかの実施形態において、ネイティブプラットフォームは、いくつかのリクエストでは抽象化プラットフォームと同一であり得、この場合、ボックス１２０４および１２０６の変換動作はスキップされ得る。

【００９２】

[0113]ボックス１２０８において、変換されたリクエストが、ネイティブプラットフォームによってリクエストを実施させるためにネイティブプラットフォームに送信される。例えば、ネイティブプラットフォームがＩｎｔｅｌ　Ｓｏｆｔｗａｒｅ　Ｇｕａｒｄ　Ｅｘｔｅｎｓｉｏｎｓ（ＳＧＸ）エンクレーブアーキテクチャに準拠する場合、ネイティブプラットフォームは、エンクレーブのためのプロセッサー命令を含み得る。この場合、ボックス１２０８においてリクエストを送信することは、エンクレーブのための１つまたは複数のプロセッサー命令を実行することを含み得る。別の例において、ネイティブプラットフォームは、エンクレーブのためのハイパーコールを有するハイパーバイザーを含み得るＭｉｃｒｏｓｏｆｔ　Ｖｉｒｔｕａｌ　Ｓｅｃｕｒｅ　Ｍｏｄｅ（ＶＳＭ）エンクレーブアーキテクチャに準拠し得る。ハイパーコールは、ハイパーバイザーコードへのソフトウェアトラップであり、ハイパーコールは、特権操作が許可され得るコンテキストへのプロセッサーコンテキストの変更を引き起こし得る。このＶＳＭ例において、ボックス１２０８においてリクエストを送信することは、特権操作が許可され得るコンテキストへ実行コンテキストを切り替えるためのハイパーバイザーおよび／または他の機序へのハイパーコールを作ることを含み得る。

【００９３】

[0114]ここでネイティブプラットフォームへリクエストを送信することは、一般には、ネイティブプラットフォームの特徴を使用してリクエストを実施することを意味する。ネイティブプラットフォーム１２０８へリクエストを送信する動作は、ネイティブプラットフォームでの複数の動作に関与し得、エンクレーブを作成すること、アテステーション、データ密封、制御転送、または単調カウンターおよび信頼できる時刻の使用など、リクエストされる動作（またはプリミティブ）によって異なり得る。

【００９４】
[0115]ＣｒｅａｔｅＥｎｃｌａｖｅプリミティブは、エンクレーブをインスタンス化するために使用され得る。エンクレーブをインスタンス化するためのＣｒｅａｔｅＥｎｃｌａｖｅリクエストは、セキュアコンテナを作成し（例えば、何らかのメモリを割り当て、そのメモリのためにセキュリティまたは隔離境界を確立することによって）、エンクレーブコードをそのセキュアコンテナ内へ（例えば、エンクレーブ画像から）コピーし、エンクレーブコード内へのエントリーポイントを構成するまたは可能にする（例えば、エンクレーブ画像内のエントリーポイントメタデータに従って）ことを、抽象化プラットフォームに行わせ得る。

【００９５】
[0116]エンクレーブ対応のハイパーバイザー（ＶＳＭなど、エンクレーブ管理機能を提供するハイパーバイザー）を有するネイティブプラットフォームにＣｒｅａｔｅＥｎｃｌａｖｅリクエストを送信することは、メモリを割り当てること、およびエンクレーブコンテナの外側のコードがそのメモリにアクセスすることを防ぐ様式でメモリのためのプロセッサーページテーブルをセットアップするためのハイパーコールを作ることを含み得る。抽象化プラットフォームからのエンクレーブ作成ハイパーコールはまた、ハイパーバイザーに、指定のエントリーポイントにおけるエンクレーブ内への制御転送のための構成情報をセットアップさせ得る。その後、セキュアコンテナの外側のコードは、セキュアコンテナの内側の指定のエントリーポイントにおける実行を転送するための制御転送ハイパーコールを作ることができる。

【００９６】
[0117]エンクレーブ対応プロセッサー（ＳＧＸなど、エンクレーブ管理プロセッサー命令を有するプロセッサー）を有するネイティブプラットフォームへのＣｒｅａｔｅＥｎｃｌａｖｅリクエストを送信することは、抽象化プラットフォームが、特定のメモリ領域がセキュアエンクレーブコンテナとして作成されるべきことをＣＰＵに通知するためにＥＣＲＥＡＴＥなどの命令を実行すること、およびそのエンクレーブコンテナにデータページを追加するためにＥＡＤＤなどの命令を実行することを含み得る。特別なプロセッサー命令はまた、エンクレーブ内への制御転送のためにエンクレーブ内のエントリーポイントを指定するためのメモリ内の特別なページを作成するために使用され得る。その後、セキュアコンテナの外側のコードは、そのエンクレーブエントリーポイントへ実行制御を転送するために指定のエントリーポイントのうちの１つを特定するＥＥＮＴＥＲなどの命令を実行することができる。

【００９７】
[0118]ＣｒｅａｔｅＲｅｐｏｒｔプリミティブは、アテステーションレポートを作成するために使用され得る。エンクレーブのアテステーションレポートを作成するためのＣｒｅａｔｅＲｅｐｏｒｔリクエストは、図５および図７に関して上に説明されるような抽象化層によって実施され得る。エンクレーブ対応のハイパーバイザーでは、抽象化層は、実行状態を、例えば、署名付きのレポートへ使用され得る図７のＰｒｉｖＡＫ７２６などの秘密鍵へのアクセスを有するセキュリティモニターコンテキストへと変更するハイパーコールを作ることによって、リクエストをネイティブプラットフォームに送信し得る。この秘密鍵は、コンピューターがＴＰＭに基づくＴＣＧログで検証されるような健全な構成で起動された場合、セキュリティモニターコンテキストにだけ利用可能であり得る。セキュリティモニターは、レポートデータに、証明されているエンクレーブのアイデンティティをタグ付けし得る。

【００９８】
[0119]エンクレーブ対応のプロセッサーでは、ＣｒｅａｔｅＲｅｐｏｒｔリクエストは、レポートを生成し、それをレポートに署名するためのプライベート鍵へのアクセスを有する特別なエンクレーブに送信するＥＲＥＰＯＲＴなどの命令を実行することによって、ネイティブプラットフォームに送信され得る。

【００９９】

[0120]ＥｎｃｌａｖｅＳｅａｌプリミティブは、エンクレーブへデータを密封するために使用され得る。エンクレーブへデータを密封することは、エンクレーブと関連付けられる様式で、またはエンクレーブと関連付けられる鍵を用いて、データを暗号化する。ＥｎｃｌａｖｅＳｅａｌリクエストは、密封ポリシーを使用して、エンクレーブの状態のすべてまたは部分など、エンクレーブの内側に位置するデータをエンクレーブへ密封するためのリクエストであり得る。上のＳＥＡＬＩＮＧ＿ＰＯＬＩＣＹなどの密封ポリシーは、どのエンクレーブがデータを開封することができるべきかを示すエンクレーブアイデンティティタイプを特定し得る。密封プロセス自体が、密封ポリシー内で特定されるエンクレーブアイデンティティと関連付けられた暗号化鍵を使用し得る。その後、特定されたアイデンティティタイプでの新しいエンクレーブのアイデンティティ値が、特定されたアイデンティティタイプでの密封エンクレーブのアイデンティティ値と同じである場合、新しいエンクレーブインスタンス化は、データを開封することが可能であり得る。

【０１００】

[0121]データ密封は、秘密または極秘エンクレーブデータが、非セキュアの記憶装置へ、例えば、エンクレーブのセキュアコンテナの外側のメモリへ、またはハードディスクなどの永続記憶装置などへ、安全にコピーされることを可能にする。密封データがエンクレーブ状態データであるとき、密封は、エンクレーブが以前の状態へとリセットされることを可能にし、またセキュアなエンクレーブ動作が中断され、その後別のエンクレーブ内で継続されることを可能にする。

【０１０１】

[0122]エンクレーブ状態をリセットするために、まずエンクレーブの状態は、その状態をエンクレーブへ密封することによって保存される。密封は、エンクレーブと関連付けられた鍵を用いて状態データを暗号化することによって行われ得る。その後、おそらくはエンクレーブの状態が変更された後、密封された状態データは、密封データを復号し、次いでエンクレーブの現在の状態を復号されたデータと置き換えることによって（例えば、復号されたデータをエンクレーブのセキュアコンテナ内へコピーすることによって）、同じエンクレーブへ開封され得る。

【０１０２】

[0123]セキュア動作を中断し、別のエンクレーブ内で継続するために、セキュア動作は、第１のエンクレーブ内に複数のプロセッサー命令を含む動作を実行することによって開始する。第１のエンクレーブが中断されるとき、そのエンクレーブの状態は、密封ポリシー内で特定されるエンクレーブアイデンティティへ密封され得、その密封データは、次いで、ローカルまたはクラウドベースの永続記憶装置など、非セキュアの記憶装置内に保存され得る。第１のエンクレーブは、次いで、（任意選択的に）終了され得るか、または他のエンクレーブ動作を開始し得る。第２のエンクレーブは、密封された状態データを第２のエンクレーブ内へ開封することによって、中断された動作を継続するためにインスタンス化またはリパーパスされ得る。中断された動作は、第１のエンクレーブが停止した場合に第２のエンクレーブ内で継続され得る。

【０１０３】

[0124]エンクレーブ対応のハイパーバイザーでは、抽象化層は、ハイパーコールを作ることによってＥｎｃｌａｖｅＳｅａｌリクエストをネイティブプラットフォームに送信し得る。ハイパーコールは、実行状態を、データを密封または開封するために使用され得るエンクレーブと関連付けられた秘密密封鍵へのアクセスを有するコンテキスト、例えば、セキュリティモニターコンテキストへ変更し得る。密封鍵は、エンクレーブアイデンティティおよびセキュリティモニターにだけ利用可能な秘密プラットフォーム鍵の組合せから導出され得る。このプラットフォーム鍵は、マシンが健全な構成で起動されるときにセキュリティモニターにのみ利用可能であり得、起動構成は、ＴＰＭに基づくＴＣＧログで検証される。このエンクレーブ対応のハイパーバイザー実施形態において、エンクレーブコードは、密封鍵へのアクセスを決して有さない。

【０１０４】

[0125]エンクレーブ対応のプロセッサーでは、ＥｎｃｌａｖｅＳｅａｌリクエストは、暗号化鍵を得るためにＥＧＥＴＫＥＹなどの命令を実行することによってネイティブプラットフォームに送信され得る。このアルゴリズムは、エンクレーブに固有である密封鍵を生成し得る。密封鍵は、エンクレーブのアイデンティティおよびプロセッサー内に埋め込まれた秘密から導出され得る。エンクレーブの内側のコードは、次いで、密封鍵を用いてデータを暗号化し得る。データは、例えば、エンクレーブの内側のコードによって、抽象化プラットフォームによって、またはネイティブプラットフォームによって、密封鍵を用いて暗号化することによって密封され得る。ＥｎｃｌａｖｅＵｎｓｅａｌは、同様に、開封鍵を生成するためにＥＧＥＴＫＥＹを使用し得る。

【０１０５】

[0126]制御転送リクエストは、エンクレーブの内側の命令からエンクレーブの外側のエントリーポイントへ（例えば、ＣａｌｌＥｎｃｌａｖｅＯｕｔ）、または逆にエンクレーブの外側の命令からエンクレーブの内側のエントリーポイントへ（例えば、ＣａｌｌＥｎｃｌａｖｅＩｎ）、プロセッサー実行制御を転送するためのリクエストであり得る。これは、例えば、セキュアなデータベース動作のために行われ得る。データベースエンクレーブをインスタンス化した後、エンクレーブクライアントは、エンクレーブが、クエリを実施するデータベースエンクレーブの内側のエントリーポイントへ制御を転送するためにＣａｌｌＥｎｃｌａｖｅＩｎプリミティブを使用することによってデータベースクエリなどの特定の動作を実施することをリクエストし得る。エンクレーブがクエリを完了した後、クエリの結果は、クエリ結果を受信し得るクライアント内のエントリーポイントにおいてクライアントに制御を転送して戻すためのＣａｌｌＥｎｃｌａｖｅＯｕｔプリミティブを用いて、（おそらくは結果を暗号化した後に）クライアントに返され得る。ＣａｌｌＥｎｃｌａｖｅＩｎおよびＣａｌｌＥｎｃｌａｖｅＯｕｔプリミティブは、エンクレーブとそのクライアントとの間で共有され得るメモリバッファへのポインターをとり得る（バッファは、エンクレーブまたはそのクライアントのいずれかによって読み込み可能、書き込み可能、および／または実行可能であり得る）。

【０１０６】

[0127]エンクレーブ対応のハイパーバイザーでは、抽象化層は、ハイパーコールを作ることによってＣａｌｌＥｎｃｌａｖｅＩｎリクエストをネイティブプラットフォームに送信し得る。ハイパーコールは、実行状態を、ＣＰＵレジスターを保存し、エンクレーブＣＰＵレジスター値の以前に保存されたセットをリストアし（おそらくはエンクレーブメモリから）、エンクレーブの保護されたメモリへのアクセスを可能にするためにページテーブル構成を変更し、エンクレーブの内側で関数エントリーポイントを呼び出すコンテキスト、例えば、セキュリティモニターコンテキストへ変更し得る。同様に、抽象化プラットフォームがＣａｌｌＥｎｃｌａｖｅＯｕｔリクエストを受信するとき、リクエストは、エンクレーブＣＰＵレジスターを保存し（おそらくはエンクレーブメモリへ保存する）、エンクレーブクライアントのための以前に保存されたＣＰＵレジスターをリストアし、エンクレーブメモリへのアクセスを防ぐためにページテーブル構成を変更し、エンクレーブの外側のエンクレーブクライアント内のエントリーポイントへプロセッサー制御を転送するハイパーコールによって、ネイティブプラットフォームへ送信され得る。

【０１０７】

[0128]エンクレーブ対応のプロセッサーでは、ＣａｌｌＥｎｃｌａｖｅＩｎリクエストは、エンクレーブＣＰＵレジスターのセットをリストアし（おそらくはエンクレーブメモリから）、エンクレーブの内側で関数を呼び出す（エントリーポイントへ制御を転送する）ことをＣＰＵに行わせ得るＥＥＮＴＥＲなどの命令を実行することによって、ネイティブプラットフォームに送信され得る。ＣａｌｌＥｎｃｌａｖｅＯｕｔプリミティブは、エンクレーブの外側の命令に制御を転送し得る、および／またはエンクレーブの外側の制御を転送する故障を引き起こし得るＥＥＸＩＴなどの命令を実行し得る。

【０１０８】

[0129]単調カウンターは、様々な用途を有し得る。例えば、エンクレーブは、その状態

がどれくらい前まで戻されるかを制限することを望み得る。単調カウンターは、図４に関して上に論じられるように、例えば、メッセージの新鮮性を保証するためのノンスとして使用され得る。単調カウンターは、一般的に、読み込まれる、および増分される能力を有するが、減少されることはできない。ロールバックまたはエンクレーブの状態を戻すことを制限するために、エンクレーブの内側のコードは、関連付けられた単調カウンターを増分し、次いでエンクレーブの内部状態と一緒にカウンターの値を保存し得る。状態およびカウンター値は、例えば、ＥｎｃｌａｖｅＳｅａｌプリミティブを用いて保存され得る。その後、例えば、ＥｎｃｌａｖｅＵｎｓｅａｌプリミティブを使用してエンクレーブ状態をリストアするとき、エンクレーブの内側のコードは、単調カウンターの現在の値を読み込み得、それを開封された状態のカウンター値と比較する。開封された状態でのカウンターの値がカウンターの現在の値よりも小さい場合、エンクレーブは、開封された状態の使用を防ぎ得る。

【０１０９】

[0130]エンクレーブ対応のハイパーバイザーでは、抽象化層は、エンクレーブに露出されるハイパーコールを作ることによって、単調カウンターを読み込むか、または増分するためにネイティブプラットフォームにリクエストを送信し得る。カウンターを読み込むか、または増分するためのハイパーコールが呼び出されると、プロセッサーは、実行状態を、セキュリティモニターなどの、ハイパーコールを作るエンクレーブのアイデンティティを検証するコンテキストへと変更し、次いで、それぞれ、例えば、ＴＰＭチップなどの不揮発性記憶装置に格納された対応する単調カウンターから読み込むか、またはこの単調カウンターを増分する。代替的に、セキュリティモニターが、リモートトラステッドサーバーまたはリモートトラステッドサーバーのセットに格納されたカウンターを、そのようなサーバーとのセキュア通信チャネルを確立し、それに特定された単調カウンターを読み込むか、または増分するように要求することによって、読み込み得るか、または増分し得る。リモートトラステッドサーバーは、エンクレーブの内側のカウンターを、ホストコンピューター内の残りのコードからそれを隔離するように維持し得る。

【０１１０】

[0131]エンクレーブ対応のプロセッサーでは、リクエストは、命令を実行することによってネイティブプラットフォームに送信され得る。そのようなプロセッサーでは、単調カウンタープリミティブは、コンピューターマザーボード内のチップ内の不揮発性メモリ記憶装置内のカウンターを読み込むか、または増分することによって、実施され得る。代替的に、これらのプリミティブはまた、エンクレーブ対応のハイパーバイザーと同様、トラステッドリムーブサーバーを使用して実施され得る。

【０１１１】

[0132]図１３は、ネイティブエンクレーブプラットフォームを抽象化する方法１３００の例示的なフローチャートを描写する。エンクレーブ抽象化プラットフォームは、ボックス１３０２において、エンクレーブまたはエンクレーブクライアントからリクエストを受信し得る。ボックス１３０４において、抽象化プラットフォームは、例えば、ネイティブプラットフォームがＳＧＸに準拠するかどうかを決定することによって、ネイティブプラットフォームがエンクレーブプロセッサー命令を含むかどうかを決定し得る。もしそうである場合、エンクレーブプロセッサー命令が、ボックス１３０６において実行される。ボックス１３０８において、抽象化プラットフォームは、例えば、ネイティブプラットフォームがＶＳＭに準拠するかどうかを決定することによって、ネイティブプラットフォームがエンクレーブハイパーコールを含むかどうかを決定し得る。もしそうである場合、ネイティブプラットフォームは、エンクレーブハイパーコールを作る。エンクレーブ命令を実行することまたはエンクレーブハイパーコールを呼び出すことからの結果は、ボックス１３１２においてクリーンアップされる。クリーンアップは、例えば、エンクレーブプロセッサー命令またはエンクレーブハイパーコールの出力パラメーターまたは例外処理を抽象化層インターフェースの形式またはプロトコルへ変換することを含み得る。変換された出力パラメーターは、次いで、ボックス１３１４において元の要求元（エンクレーブまたは

クライアント）に戻される。
抽象エンクレーブアイデンティティ
【０１１２】
[0133]図１４は、エンクレーブをインスタンス化するために使用される例示的なエンクレーブバイナリ画像を描写する。エンクレーブは、エンクレーブコンテナ１４９０などのセキュアコンテナを作成し、１つまたは複数のエンクレーブ画像の部分をコンテナ内へコピーすることによって、インスタンス化され得る。エンクレーブコンテナ１４９０は、一次エンクレーブ画像１４１０への参照により作成されている場合がある。一次画像は、他の依存エンクレーブ画像への参照を含み得る。この例では、一次画像１４１０は、それぞれ依存エンクレーブ画像１４２０および１４５０への参照として依存関係１および依存関係２を含む。画像１４２０は、画像１４３０および１４４０に対するさらなる依存関係を含む一方、画像１４５０は、画像１４６０に依存する。これらすべての画像（またはそれらの部分）がコンテナ１４９０内へコピーされると、結果として生じるエンクレーブは、インスタンス化されたエンクレーブに固有または特有であるコードおよびデータを含み得る非プラットフォーム画像１４０２と、抽象化プラットフォーム１４０４画像と、ネイティブプラットフォーム画像１４０６とを含み得る。
【０１１３】
[0134]一次画像１４１０などの各エンクレーブ画像は、ＩＤ、依存関係、コード、データ、および画像のオーサーの署名を含み得る。画像１４１０の例では、２つのＩＤ１４１０．１および１４１０．２が含まれる。これらのＩＤは、そのエンクレーブ画像を用いてインスタンス化される任意のエンクレーブを識別するために個別にまたはまとめて使用され得る、例えば、ＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、またはＡｕｔｈｏｒＩＤ値に対応する抽象アイデンティティ値を特定するＵＵＩＤであり得る。描写されるように、画像１４１０は、２つのＩＤを有するが、より少ないまたはより多くのＩＤが実現可能である。画像１４１０内のコードは、エンクレーブコンテナ１４９０をホストするコンピューターのプロセッサーにより実行可能なバイナリ命令であり得る。画像１４１０内のデータは、コンテナ１４１０内へロードされる任意のコードによって使用され得る。画像１４１０はまた、ＩＤ、依存関係リファレンス、コード、およびデータなどの画像の他のコンテンツのいずれかまたはすべての完全性を確実にするために署名Ｓｉｇ１４１０を含み得る。他の画像１４２０～１４６０は同様に、ＩＤ、依存関係リファレンス、コード、データ、および署名を含み得る。
【０１１４】
[0135]依存１および依存２または画像１４１０、画像１４２０の依存１および依存２、ならびに画像１４５０の依存関係１などの依存関係指標は、様々なやり方で特定され得る。画像１４１０～１４６０は、コンピューターシステムのメモリに格納される場合、依存関係指標は、単純にメモリ内のアドレスであり得る。エンクレーブ画像がファイルシステム内のファイルである場合、リファレンスは、ファイル名であり得る。いくつかの実施形態において、リファレンスは、論理識別子であり得る。論理識別子は、ＵＵＩＤなどの固有の数字であり得るか、または、依存関係画像を別の方法で識別する文字列などの他のデータであり得る。例えば、文字列は、依存バイナリ画像のオーサー、ソース、製品名、製品ファミリー、および／またはバージョン番号を示し得る。論理識別子は、依存バイナリのコピーが取得され得る場所など、ウェブまたはインターネットの場所を含む。
【０１１５】
[0136]いくつかの実施形態において、エンクレーブ画像ファイルは、参照されるエンクレーブ画像の現在のバージョンまたはローカルコピーへのポインターを探すためにエンクレーブ画像のレジストリー内の論理識別子などの依存関係指標を検索することによって位置特定され得る。いくつかの場合においては、トラステッドサービスが、特定のエンクレーブ画像または画像位置の識別へと依存関係指標を解決するために使用され得る。
【０１１６】
[0137]いくつかの実施形態において、依存関係指標は、目的とする依存エンクレーブバ

イナリ画像の暗号学的ハッシュなど、暗号学的にセキュアな識別子であり得る。そのようなハッシュは、依存バイナリのすべて、またはその一部分のみを含み得る。依存関係指標に含まれる依存バイナリの一部分は、ＩＤ１４１０．１またはＩＤ１４２０．２などの抽象アイデンティティ値を含み得、また抽象アイデンティティ値であり得る。エンクレーブ依存関係を決定するエンティティは、正しい依存画像が依存バイナリ自体のハッシュをコンピューティングすることによって見つけられたことを検証することができる場合があるため、暗号学的にセキュアな識別子のための解決サービスは、論理識別子と同じように信頼される必要がない場合がある。

【０１１７】

[0138]図１５は、抽象エンクレーブアイデンティティを用いてエンクレーブ動作を実施する方法１５００の例示的なフローチャートを描写する。エンクレーブのための抽象アイデンティティ値は、何らかの特徴を共通して有するが正確には同一ではない２つのエンクレーブ間の等価性を決定するための基礎を提供し得る。アイデンティティ値は、アテステーションレポートに含まれ得、抽象アイデンティティタイプ（そのようなＥｘａｃｔＨａｓｈ、ＩｎｓｔａｎｃｅＨａｓｈ、ＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、またはＡｕｔｈｏｒＩＤ）と関連付けられ得る。正確には同じではない２つのエンクレーブは、抽象アイデンティティタイプについて同じ抽象アイデンティティ値を有し得る。加えて、２つの異なるネイティブエンクレーブプラットフォーム上でセキュアコンテナ内へインスタンス化される同一のエンクレーブコードもまた、同じ抽象アイデンティティ値を有し得る。方法１５００は、例えば、ネイティブエンクレーブプラットフォームとエンクレーブまたはエンクレーブクライアントのいずれかとの間の抽象化プラットフォーム層によって実施され得る。

【０１１８】

[0139]ボックス１５０２において、エンクレーブは、図１４の一次エンクレーブ画像１４１０などのエンクレーブ画像からインスタンス化される。エンクレーブ画像は、エンクレーブコード、データ、アイデンティティのリスト、依存エンクレーブ画像のリスト、および署名を含む一次画像であり得る。エンクレーブ画像の完全性を確実にするために、画像は、エンクレーブ画像のオーサーに対応し得るプライベート鍵を用いて署名され得る。エンクレーブ画像内のエンクレーブアイデンティティＩＤのリストは、ＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、およびＡｕｔｈｏｒＩＤなどの抽象アイデンティティタイプに対応し得、各々が、他の関連エンクレーブ画像と一緒に、エンクレーブ画像をまとめて識別することを目的とされる。依存エンクレーブ画像のリストは、一次エンクレーブ画像内のコードが依存するエンクレーブコードを含む他のエンクレーブ画像を指し得る。依存エンクレーブ画像は、同じオーサーによって著作される場合とそうでない場合とがあり、いくつかの依存エンクレーブ画像は、一次エンクレーブ画像または一次エンクレーブ画像のオーサーと特に関連付けられるのではなく、概してエンクレーブプラットフォーム（抽象化プラットフォームまたはネイティブプラットフォームのいずれか）と関連付けられ得る。エンクレーブは、任意のネイティブエンクレーブプラットフォームを使用してセキュアエンクレーブコンテナを作成し、一次画像のすべてまたは一部分および任意の依存エンクレーブ画像をセキュアコンテナ内へコピーすることによって、インスタンス化され得る。

【０１１９】

[0140]ボックス１５０３において、エンクレーブ動作は、エンクレーブアイデンティティティタイプと一緒に、例えば、エンクレーブまたはエンクレーブクライアントによってリクエストされる。アイデンティティタイプは、ＩｍａｇｅＩＤまたはＡｕｔｈｏｒＩＤなどの抽象アイデンティティのタイプを特定し得、特定のインスタンス化されたエンクレーブに関連し得るが、そのエンクレーブのＡｕｔｈｏｒＩＤ値を特定しない。ボックス１５０３に続く方法１５００の残りの部分は、インスタンス化されたエンクレーブのそのアイデンティティタイプのために導出されたアイデンティティ値を使用して、インスタンス化されたエンクレーブを用いて動作（アテステーション、データ密封、または単調カウンターの使用など）を実施するための動作を説明する。アイデンティティは、エンクレーブ画像

のサブセットのハッシュを使用して決定され得る。エンクレーブ画像のどのサブセットがハッシュへの入力として使用されるかは、エンクレーブ動作において使用されることが望まれるアイデンティティタイプに部分的に基づき得る。

【０１２０】

[0141]ボックス１５０４において、本明細書ではアイデンティティ部と呼ばれるエンクレーブ画像の一部分は、アイデンティティタイプに基づいて決定される。アイデンティティ部は、ボックス１５０２においてエンクレーブをインスタンス化するために使用される様々なエンクレーブバイナリ画像のすべてを含み得る、部分を含み得る、またはいずれも含まない場合がある。アイデンティティ部は、エンクレーブ画像内に含まれるエンクレーブコードのすべてを含み得る、部分を含み得る、またはいずれも含まない場合がある。アイデンティティ部はまた、含まれるエンクレーブ画像の非コード部内にリストされる、ゼロ、１つ、またはそれ以上のアイデンティティＩＤを含み得る。アイデンティティ部はまた、エンクレーブ画像に含まれるエンクレーブデータを含む場合とそうでない場合とがある。アイデンティティ部は、エンクレーブ画像のこれらの様々な部分の任意の組合せを含み得る。例えば、アイデンティティ部は、すべてのコードを含み、いずれのデータも含まず、２つまたは４つの利用可能なアイデンティティＩＤを含む場合がある。任意選択のボックス１５０６において、どの依存エンクレーブ画像がアイデンティティ部に含まれるべきかが決定され、またそれぞれの含まれる画像のアイデンティティ部が決定される。

【０１２１】

[0142]依存画像のアイデンティティ部は、一次エンクレーブ画像のアイデンティティ部と同じ場合とそうでない場合とがある。例えば、すべてのコードおよびＩｍａｇｅＩＤが、一次画像のアイデンティティ部に含まれる一方、依存画像のアイデンティティ部内には、コードは含まれず、依存画像のＦａｍｉｌｙＩＤだけが含まれ得る。

【０１２２】

[0143]エンクレーブコードがアイデンティティ部に含まれるとき、アイデンティティ部内のエンクレーブコードの一部分は、アイデンティティタイプの組合せ、およびどの依存関係がアイデンティティ部に含まれるべきかの標示によって決定され得る。アイデンティティタイプＩｎｓｔａｎｃｅＨａｓｈは、例えば、一次画像内にエンクレーブコードを含み得るが、依存画像には含まず、アイデンティティタイプＥｘａｃｔＨａｓｈは、エンクレーブプラットフォームの部分と見なされないすべての依存画像内にエンクレーブコードを含み得る。例えば、エンクレーブプラットフォームオーサーのプライベート鍵を用いて署名されないすべての依存エンクレーブ画像は、エンクレーブプラットフォームの部分ではないと見なされ得る。代替的に、または加えて、一次画像は、どの依存エンクレーブ画像が、ＩｎｓｔａｎｃｅＨａｓｈまたはＥｘａｃｔＨａｓｈアイデンティティタイプのためのアイデンティティ部に含まれるべきか、または除外されるべきかのリストを含み得る。

【０１２３】

[0144]エンクレーブ画像にメタデータとして含まれ得るエンクレーブアイデンティティＩＤは、エンクレーブコードの代わりに、またはそれに加えて、エンクレーブ画像のアイデンティティ部に含まれ得る。例えば、アイデンティティタイプＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、およびＡｕｔｈｏｒＩＤのためのアイデンティティ部は、エンクレーブ画像からの対応するＩＤメタデータを含み得る。アイデンティティタイプがネスト化または層化されるとき、より低いレベルタイプのアイデンティティ部は、より高いレベルタイプのＩＤデータを含み得る。例えば、ＩｍａｇｅＩＤのためのアイデンティティ部は、ＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、およびＡｕｔｈｏｒＩＤのためのＩＤデータを含み得る一方、ＡｕｔｈｏｒＩＤのためのアイデンティティ部は、ＡｕｔｈｏｒＩＤのためのＩＤデータのみを含み得る。

【０１２４】

[0145]ＩｎｓｔａｎｃｅＨａｓｈおよびＥｘａｃｔＨａｓｈなどのエンクレーブコードを含むアイデンティティタイプは、特定のエンクレーブコードがエンクレーブの内側で実

行しているという、例えば、アテステーションによるエンクレーブクライアントへのより高いレベルの確証を提供する。しかしながら、エンクレーブのアイデンティティは、エンクレーブコードのアイデンティティ部のいずれかが変化するときに必ず変化することになる。例えば、セキュリティ処置または他のバグがエンクレーブ画像の新しいバージョンで修正される場合、新しいコードに基づく結果として生じるアイデンティティ値も変化することになる。エンクレーブコードの特定の部分がアイデンティティハッシュ計算から除外される機序を提供することによって、エンクレーブのアイデンティティは、エンクレーブコードの除外された部分への変更から切り離され得る。例えば、あるオーサーのエンクレーブコードがエンクレーブプラットフォームによって提供されるエンクレーブコードに依存するとき、エンクレーブアイデンティティは、依存プラットフォームへの改正から切り離され得る。

【０１２５】

[0146]ボックス１５０８において、ボックス１５０２においてインスタンス化されるエンクレーブのアイデンティティを表し得るアイデンティティ値が決定される。アイデンティティ値は、エンクレーブ画像の以前に決定されたアイデンティティ部についてハッシュを計算することによって決定され得る（アイデンティティ値は、アイデンティティ部がハッシュ関数への入力である場合、ハッシュ関数の出力である）。いくつかの実施形態において、ハッシュ関数への入力は、元のエンクレーブ画像の一部分であるが、他の実施形態において、ハッシュ関数への入力は、画像のアイデンティティ部がコンテナ内へコピーされた（およびおそらくは、元のエンクレーブ画像が暗号化される場合には、アイデンティティ部を復号した）後のエンクレーブコンテナの一部分である。

【０１２６】

[0147]ボックス１５１０において、結果として生じるアイデンティティ値の完全性は、元のエンクレーブ画像の完全性を検証することによって、任意選択的に検証され得る。エンクレーブ画像の完全性は、エンクレーブ画像に署名するために使用されるプライベート鍵に対応する公開鍵を用いて検証され得る。そのような公開／プライベート鍵ペアは、例えば、エンクレーブ画像のオーサーと関連付けられ得るため、結果として生じるアイデンティティ値における信頼は、エンクレーブのオーサーの信頼に根差し得る。

【０１２７】

[0148]最後に、ボックス１５１２において、インスタンス化されたエンクレーブに関する動作が、アイデンティティ値を使用して実施され得る。例えば、インスタンス化されたエンクレーブのアテステーションレポートが、アイデンティティタイプについて生成または検証され得、データが、アイデンティティにおいて、インスタンス化されたエンクレーブへ密封され得るか、またはそこから開封され得、ならびに、インスタンス化されたエンクレーブおよびアイデンティティタイプに結び付けられた単調カウンターまたは信頼できる時刻が使用され得る。

【０１２８】

[0149]より高いレベルのアイデンティティタイプを使用したエンクレーブ動作は、可能性のあるエンクレーブインスタンス化のグループ間の相互作用を可能にする。高レベルアイデンティティタイプへのアテステーションは、同じ高レベルアイデンティティを有するすべてのエンクレーブのためのアテステーションレポート等価性を提供し得る。例えば、ＡｕｔｈｏｒＩＤアイデンティティタイプへのアテステーションレポートは、同じＡｕｔｈｏｒＩＤメタデータを含む一次画像からインスタンス化されるすべてのエンクレーブからのアテステーションレポートに等価であり得る。高レベルアイデンティティタイプへ密封されるデータは、同じ高レベルアイデンティティ値を有する任意のエンクレーブによって開封され得る。例えば、ＡｕｔｈｏｒＩＤアイデンティティタイプを有するインスタンス化されたエンクレーブへ密封されるデータは、そのエンクレーブ一次画像内の同じＡｕｔｈｏｒＩＤメタデータを有する任意の他のインスタンス化されたエンクレーブによって開封され得る。

エンクレーブアイデンティティ等価性

[0150]図１６は、抽象エンクレーブアイデンティティ等価性を有する例示的なシステムを描写する。エンクレーブクライアント１６０２は、抽象化プラットフォーム１６１４を介して、第１のネイティブプラットフォーム１６１６のセキュアエンクレーブコンテナ内でインスタンス化される第１のエンクレーブ１６１２と通信し得、クライアント１６０２はまた、抽象化プラットフォーム１６２４を介して、第２のネイティブプラットフォーム１６２６のセキュアエンクレーブコンテナ内でインスタンス化される第２のエンクレーブ１６２２と通信し得る。第１のネイティブプラットフォーム１６１６は、第２のネイティブプラットフォームと同じコンピューター上に存在する場合とそうでない場合とがある。エンクレーブクライアント１６０２は、ネイティブプラットフォームのいずれかと同じコンピューター上に存在し得るか、または別個の第３のコンピューター上に存在し得る。第１のネイティブプラットフォーム１６１６は、第２のネイティブプラットフォーム１６２６と同じではない場合がある。例えば、第１のネイティブプラットフォーム１６１６は、同じネイティブプラットフォーム製造業者からの第２のネイティブプラットフォームの古いバージョンであり得る。代替的に、第１のネイティブプラットフォーム１６１６および第２のネイティブプラットフォーム１６２６は、ＶＳＭおよびＳＧＸなど、完全に異なるエンクレーブアーキテクチャに準拠し得る。

【０１２９】

[0151]エンクレーブクライアントは、アテステーションレポートから導出されるアイデンティティ値を比較することによってエンクレーブが等価であることをセキュアに決定し得る。エンクレーブクライアント１６０２は、第１のエンクレーブ１６１２および第２のエンクレーブ１６２２から別個のアテステーションレポートを受信することによってエンクレーブの各々をセキュアに識別し得る。アイデンティティ値は、これらのアテステーションレポートの各々に含まれ得る（またはそれから導出され得る）。アイデンティティ値が同じである場合、エンクレーブクライアント１６０２は、第１のエンクレーブ１６１２および第２のエンクレーブ１６２２がある意味では等価であることに自信を有し得る。アテステーションレポートからのアイデンティティ値は、特定の抽象アイデンティティタイプ（そのようなＥｘａｃｔＨａｓｈ、ＩｎｓｔａｎｃｅＨａｓｈ、ＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、またはＡｕｔｈｏｒＩＤ）に対応する抽象アイデンティティ値、またはそのような抽象アイデンティティ値のハッシュであり得る。この場合、等価性は、エンクレーブが正確には同一ではない場合に決定され得る。２つのエンクレーブは、正確には同一ではない場合があるが、例えば、エンクレーブコンテナ内へロードされるエンクレーブ画像が、同じ機能性の異なるバージョンであるか、または異なる依存画像を有する同じ一次画像であるか、または異なるエンクレーブアーキテクチャのエンクレーブコンテナ内へロードされる同じエンクレーブ画像である場合には等価であると依然として決定され得る。

【０１３０】

[0152]第１のエンクレーブ１６１２は、様々な状況では第２のエンクレーブ１６２２に等価であるが同一ではないと見なされ得る。第１の例では、エンクレーブコンテナ内へ最初にロードされるコードのサブセットのみが同じ（例えば、抽象アイデンティティタイプＥｘａｃｔＨａｓｈまたはＩｎｓｔａｎｃｅＨａｓｈについて等価）である。第２の例では、エンクレーブコードのオーサーは、２つのバイナリ画像内のコードが異なるとしても、２つの異なるエンクレーブバイナリ画像内に同一のＩＤを含んでいる場合がある（例えば、アイデンティティタイプＩｍａｇｅＩＤ、ＦａｍｉｌｙＩＤ、またはＡｕｔｈｏｒＩＤについて等価）。第３の例では、各エンクレーブ内のコードは、全く同じであるが、異なるネイティブプラットフォーム上にロードされる（インスタンス化される）。この第３の例では、第１のネイティブプラットフォーム１６１６および第２のネイティブプラットフォーム１６２６は、異なる製造業者によって製造され得、したがって異なるアテステーションレポートの信頼は、異なる製造業者の異なる認証局に根差す（図７、要素７３８を参照のこと）。２つのネイティブプラットフォームが異なる例は、サーバーファーム内またはクラウドコンピューティング内であり、ここでは第１および第２のエンクレーブの処理作業負荷に割り当てられるサーバーは、同じネイティブエンクレーブプラットフォーム

をサポートしないサーバーである。

【０１３１】

[0153]代替的な実施形態において、第１のエンクレーブは、第２のエンクレーブのクライアントであり得るため、結果として、ボックス１６０２および１６１２は組み合わされる。この実施形態においてエンクレーブ等価性を決定することは、第１のエンクレーブ内で、第２のエンクレーブのアテステーションレポートからのアイデンティティ値が第１のエンクレーブの独自のアイデンティティ値と同じである（特定の抽象アイデンティティレベルで）ことを決定することを含み得る。

【０１３２】

[0154]図１７は、２つの等価のエンクレーブを用いた並列処理の例示的なフローチャートを描写する。プロセス１７００は、例えば、２つ以上の異なるエンクレーブのクライアントによって実施され得る。ボックス１７０２において、２つのエンクレーブは、例えば図１６に描写されるように、異なるネイティブプラットフォームインスタンス上でインスタンス化される。エンクレーブは、抽象化プラットフォーム１６１４および１６２４のＣｒｅａｔｅＥｎｃｌａｖｅ方法によりエンクレーブバイナリ画像（図１４の一次画像１４１０など）を特定するエンクレーブクライアントによってインスタンス化され得る。２つのエンクレーブをインスタンス化するために特定されるエンクレーブバイナリ画像は、同じ、または異なる場合がある。それぞれのインスタンス化されたエンクレーブのためのアテステーションレポートは、ボックス１７０４において作成される。アテステーションレポートは、例えば、エンクレーブクライアントのリクエストで、またはエンクレーブ自身のリクエストで作成され得る。エンクレーブクライアントなどの２つのエンクレーブの等価性を証明することを望むエンティティは、両方のアテステーションレポートのコピーを得る。アテステーションレポートは、ボックス１７０６において任意選択的に検証され得る。例えば、レポートの完全性は、アテステーションレポートをもたらしたネイティブプラットフォームの承認証明書（図７、要素７２８など）を用いてアテステーション署名を検証することによって検証され得る。さらに、承認証明書は、ネイティブプラットフォーム製造業者の公開鍵（図７、要素７３２など）を用いて検証され得る。アイデンティティ値（またはそのハッシュ）は、ボックス１７０８において各アテステーションレポートから抽出され得、２つのエンクレーブの等価性は、抽出されたアイデンティティ値が各エンクレーブで同じであることを検証することによって決定され得る。これらのアイデンティティ値は、アイデンティティタイプと関連付けられた抽象アイデンティティ値（またはそのハッシュ）であり得る。

【０１３３】

[0155]エンクレーブクライアントがボックス１７０８および１７１０における動作から２つのエンクレーブインスタンス化の等価性を証明した後は、２つのエンクレーブは、示される等価性のタイプによって、区別なしに使用され得る。ボックス１７１２～１７２０は、２つのインスタンス化された等価のエンクレーブを並列処理様式で使用するための等価のエンクレーブを使用する例示的な方法を描写する。ボックス１７１２および１７１６において、データベースの一部分またはデジタル画像ファイルの一部分など、入力データセットの一部分が、第１および第２のエンクレーブ内へコピーされる。コピーされるデータセットの一部分は、目の前にある処理タスクによって同一であるか、または異なる場合がある。処理動作は、同時に、ボックス１７１４において第１のエンクレーブ内での動作を部分的に実施すること、およびボックス１７１８において第２のエンクレーブ内での動作を部分的に実施することによって、並列でセキュアに実施され得る。動作は、例えば、データベースを検索すること、または画像処理動作を実施することであり得る。第１のエンクレーブは、データベースの第１の半分を検索し得るか、または画像の第１の半分に対して画像処理動作を実施し得る一方、第２のエンクレーブは、データベースの第２の半分を検索し得るか、または画像の第２の半分に対して画像処理動作を実施し得る。最後に、ボックス１７２０において、第１および第２のエンクレーブ内での並列処理の結果は、例えば、データベースの２つのソートされた半分を組み合わせること、または２つの画像半

分をつなぎ合わせることによって、組み合わされ得る。

【０１３４】

[0156]図１８は、２つの等価のエンクレーブを用いた逐次処理の例示的なフローチャートを描写する。図１８に描写されるように、データベース動作または画像処理動作などのエンクレーブ動作は、２つの別個のエンクレーブ内で２つの連続的部分においてセキュアに行われる。プロセス１８００は、例えば、図１６のエンクレーブクライアント１６０２によって実施され得る。ボックス１８０２において、第１のエンクレーブが、第１のネイティブエンクレーブプラットフォーム上で作成され、第１のエンクレーブのアテステーションレポートが、ボックス１８０４において作成される。この第１のアテステーションレポート（第１のエンクレーブの）は、例えば、図１７のボックス１７０６に関して上に説明されるように、ボックス１８０６において検証され得る。ボックス１８０８において、セキュア動作が、第１のエンクレーブ内で開始されるが、完了はされない。第１のエンクレーブの状態は、ボックス１８１０において、第１のエンクレーブから外へ安全に移動されるように任意選択的に密封され得る。例えば、第１のエンクレーブ状態は、第１のエンクレーブのアイデンティティタイプへ密封され得る。エンクレーブ状態が一旦保存されると、第１のエンクレーブは終了され得る（描かれていない）。

【０１３５】

[0157]第２のエンクレーブは、ボックス１８１２において開始して使用される。ボックス１８１２において、第２のエンクレーブは、第２のネイティブプラットフォーム上でインスタンス化される。図１６および図１７にあるように、第２のネイティブプラットフォームは、第１のネイティブプラットフォームと同じコンピューター上でホストされる場合とそうでない場合とがあり、第１および第２のネイティブプラットフォームは、同じであるか、または異なる場合がある。第２のネイティブプラットフォームのアテステーションレポートは、ボックス１８１４において作成され、また、任意選択的に、この第２のアテステーションレポートは、ボックス１８１６において検証され得る。第１および第２のアテステーションレポートからのアイデンティティ値は、第１および第２のエンクレーブの等価性を検証するために、ボックス１８１８において比較され得る。代替的な実施形態において、第２のエンクレーブは、セキュア動作がボックス１８０８において第１のエンクレーブ内で開始される前に、インスタンス化されかつ等価性検証され得る（ボックス１８１２～１８１８）。第１のエンクレーブ内で開始されるセキュア動作を継続するために、第１のエンクレーブからの密封状態は、ボックス１８２０において第２のエンクレーブ内へコピーされ開封され得る。ボックス１８２２において、セキュア動作は、第２のエンクレーブ内で完了される（状態がコピーされた場合は、第１のエンクレーブからセキュアにコピーされるエンクレーブ状態を使用して）。

【０１３６】

分散データ密封

[0158]図１９は、例示的な分散データ密封システムのブロック図である。データ密封は、複数のエンクレーブにわたって分散され得、この場合、これらのエンクレーブは、別個のネイティブエンクレーブプラットフォーム上、および／または別個のコンピューター上でホストされる。上に説明されるように、ＥｎｃｌａｖｅＳｅａｌおよびＥｎｃｌａｖｅＵｎｓｅａｌ抽象化プリミティブは、エンクレーブが実行しているネイティブエンクレーブプラットフォームまたは物理的なコンピューターに結び付けられる鍵を使用してエンクレーブのためのデータを密封および開封し得る。これは、同じコンピューターまたは同じネイティブエンクレーブプラットフォームインスタンス上でホストされるエンクレーブにのみ開封を制限し得る。図１９は、データを密封または開封することが、エンクレーブをホストするネイティブプラットフォームおよびコンピューターとは異なるネイティブプラットフォームまたはコンピューター上で発生し得る分散データ密封システムを描写する。システム１９００は、コンピューター１９１０、１９３０、１９５０を、コンピューター１９１０および１９３０を接続するネットワーク１９０２、ならびにコンピューター１９３０および１９５０を接続するネットワーク１９０４と共に含む。コンピューター１９１

０は、発信元エンクレーブ１９１２をホストし、この発信元エンクレーブ１９１２から密封されるべきデータが供給され得、コンピューター１９３０は、分散密封および開封リクエストをサービスするための分散密封エンクレーブ（ＤＳＥ）１９３２をホストし、コンピューター１９５０は、以前に密封されたデータが開封される宛先エンクレーブ１９５２をホストする。図９に関して上に説明されるように、エンクレーブ１９１２、１９３２、１９５２は、エンクレーブ抽象化プロトコルを介して、抽象化プラットフォーム１９１４、１９３４、１９５４と通信し得、抽象化プラットフォーム１９１４、１９３４、１９５４は、ネイティブプロトコルを介して、ネイティブプラットフォーム１９１６、１９３６、および１９５６とそれぞれ通信し得る。代替的な実施形態において、１つまたは複数のエンクレーブ１９１２、１９３２、１９５０は、中間の抽象化プラットフォームなしにネイティブプラットフォーム１９６１、１９３６、１９５６上で直接的にホストされ得る。密封データ１９３８は、ＤＳＥ１９３２またはそのホスティングネイティブプラットフォーム１９３６と関連付けられた鍵を使用してＤＳＥ１９３２へ密封されるデータであり得る。密封データ１９３８は、ＤＳＥ１９３２のセキュアエンクレーブコンテナの外側のコンピューター１９３０上など、あまり保護されていない場所、例えば、コンピューター１９３０のメモリ空間内またはハードディスクのファイルシステム内の他の場所に格納され得る。

【０１３７】

[0159]分散データ密封は、例えば、ネットワーク１９０２を介したＤＳＥ１９３２のアテステーションによる、発信元エンクレーブへのＤＳＥ１９３０の認証を含み得る。発信元エンクレーブ１９１２が一旦ＤＳＥ１９３２を信頼すると、発信元エンクレーブ１９１２は、セキュア通信チャネルを介してＤＳＥ１９３２による密封のための密封ポリシーと一緒に極秘データをＤＳＥ１９３２に送信し得る。ＤＳＥ１９３２は、次いで、エンクレーブ１９１２からのデータをそれ自体の中に密封し、その密封データを非セキュアの記憶装置に格納し得る。その後、宛先エンクレーブ１９５２は、以前に密封されたデータをリクエストし得る。データを開封するために、ＤＳＥ１９３２は、宛先エンクレーブ１９５２を、例えば、ネットワーク１９０４を介したアテステーションによって認証し、宛先エンクレーブ１９５２のための開封が発信元エンクレーブ１９１２によって提供される密封ポリシーに従って許可されることを検証し得る。ＤＳＥ１９３２は、発信元エンクレーブ１９１２からの以前に密封されたデータを開封し、次いで開封されたデータをセキュア通信チャネルを介して宛先エンクレーブ１９５２に送信し得る。エンクレーブデータは、ネットワーク１９０２および１９０４を介して、エンクレーブデータを暗号化することによってＤＳＥ１９３２へおよびＤＳＥ１９３２からセキュアに通信され得る。例えば、ネットワーク１９０２を介して送信されるエンクレーブデータは、発信元エンクレーブ１９１２へのＤＳＥ１９３２のアテステーション中に生成される鍵を用いて暗号化され得、ネットワーク１９０４を介して送信されるデータは、ＤＳＥ１９３２への宛先エンクレーブ１９５２のアテステーション中に生成される鍵を用いて暗号化され得る。宛先の公開鍵、例えば、ＤＳＥと関連付けられた公開鍵、または宛先エンクレーブと関連付けられた公開鍵を用いた暗号化など、他のセキュア通信チャネルが可能である。

【０１３８】

[0160]分散密封および開封で使用されるエンクレーブアイデンティティは、抽象エンクレーブアイデンティティである場合とそうでない場合とがある。例えば、抽象化プラットフォーム層を用いたいくつかの実施形態において、発信元エンクレーブによって特定され、ＤＳＥによって実行されるものなどの密封ポリシーは、許可された開封エンクレーブアイデンティティを識別し得、ここでは、許可された開封エンクレーブアイデンティティは、例えば、抽象エンクレーブアイデンティティのリスト、または発信元エンクレーブの抽象アイデンティティ値と組み合わせた抽象アイデンティティタイプのリストである。他の状況において、非抽象アイデンティティが使用され得る。例えば、いくつかの実施形態において、ＤＳＥは、一般に利用可能なコードで実施され得るため、結果としてＤＳＥにおける信頼は、そのコードのオーサーにおける信頼に対立するものとして、そのコードの知

識における信頼である。この例では、ＤＳＥのアテステーションは、ＤＳＥの公開コードのすべての署名付きハッシュであり得、ハッシュ関数への入力は、オーサーによって指定された抽象アイデンティティ値を含まない場合がある。

【０１３９】

[0161]いくつかの実施形態において、ネイティブプラットフォーム１９１６、１９３６、１９５６が同じネイティブエンクレーブプラットフォームアーキテクチャの同じバージョンに準拠するとしても、それらが異なるコンピューター上１９１０、１９３０、１９５０でホストされることから、ネイティブプラットフォーム１９１６、１９３６、１９５６は、別個のネイティブプラットフォームである。他の実施形態において、ネイティブプラットフォーム１９１６、１９３６、１９５６は、異なるプラットフォームアーキテクチャ、または同じネイティブエンクレーブプラットフォームアーキテクチャの異なるバージョンに準拠し得る。密封ポリシー内の抽象アイデンティティの使用は、発信元および宛先エンクレーブを異なるネイティブプラットフォームアーキテクチャ上でホストすることを促進し得る。

【０１４０】

[0162]図１９に描かれていない分散データ密封の他の実施形態において、３つの別個のコンピューター（別個のコンピューター１９１０、１９３０、１９５０など）が存在しない場合がある。例えば、発信元および宛先エンクレーブは、１つのコンピューター上（およびおそらくは単一のネイティブプラットフォーム上）にあり得る一方、ＤＳＥは、別個のコンピューター上にあり得る。代替的に、ＤＳＥは、発信元エンクレーブをホストするコンピューターまたは宛先エンクレーブをホストするコンピューターのいずれかと同じコンピューター上でホストされ得る。これらの分散データ密封実施形態において、データ密封および開封は、ＥｎｃｌａｖｅＳｅａｌおよびＥｎｃｌａｖｅＵｎｓｅａｌ抽象化プリミティブに関して上に説明されるように、単一のコンピューターに完全にはローカルではない。

【０１４１】

[0163]分散データ密封は、抽象化プラットフォーム１９１４、１９３４、１９５４によってなど、抽象化層ＡＰＩ内で実施され得る。例えば、ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌおよびＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＵｎｓｅａｌプリミティブは、上に論じられるローカルデータ密封プリミティブＥｎｃｌａｖｅＳｅａｌおよびＥｎｃｌａｖｅＵｎｓｅａｌに類似する。

```
ＤＷＯＲＤ
ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌ（
＿Ｉｎ＿　ＳＥＡＬＩＮＧ＿ＰＯＬＩＣＹ　ｓｅａｌｉｎｇＰｏｌｉｃｙ，
＿Ｉｎ＿ｒｅａｄｓ＿ｂｙｔｅｓ＿ｏｐｔ＿（ｄｗＰｌａｉｎｔｅｘｔＳｉｚｅ）ＬＰＣＶＯＩＤ　ｐＰｌａｉｎｔｅｘｔ，
＿Ｉｎ＿　ＤＷＯＲＤ　ｄｗＰｌａｉｎｔｅｘｔＳｉｚｅ，
＿Ｉｎ＿ｒｅａｄｓ＿ｂｙｔｅｓ＿ｏｐｔ＿（ｄｗＡｕｔｈｄａｔａＳｉｚｅ）ＬＰＣＶＯＩＤ　ｐＡｕｔｈｄａｔａ，
＿Ｉｎ＿　ＤＷＯＲＤ　ｄｗＡｕｔｈｄａｔａＳｉｚｅ，
＿Ｏｕｔ＿ｗｒｉｔｅｓ＿ｂｙｔｅｓ＿ｔｏ＿（ｄｗＳｅａｌｅｄｔｅｘｔＳｉｚｅ）ＬＰＶＯＩＤ　ｐＳｅａｌｅｄｔｅｘｔ，
＿Ｉｎｏｕｔ＿　ＤＷＯＲＤ　ｄｗＳｅａｌｅｄｔｅｘｔＳｉｚｅ，
Ｓｅｔ＜ＥｎｃｌａｖｅＩｄｅｎｔｉｔｙ＞ＳｅｔＯｆＴａｒｇｅｔＥｎｃｌａｖｅｓ
）
```

【０１４２】

[0164]ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌは、エンクレーブ１９１０などの呼び出しエンクレーブが、ｐＰｌａｉｎｔｅｘｔパラメーターにより提供されるデータを開封する権限を与えられるエンクレーブアイデンティティのセットを特定することを可能にする追加のＳｅｔＯｆＴａｒｇｅｔＥｎｃｌａｖｅｓパラメーターをとることにより

、ＥｎｃｌａｖｅＳｅａｌを拡張する。ＳｅｔＯｆＴａｒｇｅｔＥｎｃｌａｖｅｓにより提供されるアイデンティティがない場合、デフォルトで権限を与えられたエンクレーブアイデンティティが、密封エンクレーブのアイデンティティ、例えば、密封エンクレーブのＥｘａｃｔＨａｓｈまたはＩｎｓｔａｎｃｅＨａｓｈであると仮定され得る。

【０１４３】

[0165]ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌの実装形態は、例えば、発信元エンクレーブのコンピューター上の抽象化プラットフォーム１９１４の方法のように、ネットワーク１９０２を介してメッセージを暗号化することなどによって、ＤＳＥとのセキュア通信チャネルを確立することを含み得る。この暗号化のための鍵は、例えば、上に説明されるように、アテステーションプロセス中に生成され得るか、または、ＤＳＥ１９３２と関連付けられた任意の公開鍵であり得る。

【０１４４】

[0166]ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌは、追加のパラメーターＫｅｙＦｏｒＤａｔａ（上のＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌ関数プロトタイプでは示されない）をとることによりさらに一般化され得る。いくつかの実施形態において、データの複数のセットが、単一のエンクレーブまたは単一のエンクレーブアイデンティティのために同時に密封したままに保たれ得る。この場合、ＫｅｙＦｏｒＤａｔａは、データのどのセットが密封されているかの特定を可能にする。ＫｅｙＦｏｒＤａｔａは、文字列、数字、またはプロパティのセットなど、あらゆる種類のデータ識別子であり得る。いくつかの実施形態において、ＫｅｙＦｏｒＤａｔａは、ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌへの入力パラメーターであり得、密封エンクレーブによって生成され得、密封エンクレーブがデータセットを命名することを効果的に可能にする。他の実施形態において、ＫｅｙＦｏｒＤａｔａは、出力パラメーターであり得、ここではＤＳＥは、データが密封されると、ＫｅｙＦｏｒＤａｔａ識別子を生成する。

```
DWORD
DistributedEnclaveUnseal(
_In_reads_bytes_opt_(dwSealedtextSize) LPCVOID pSealedtext,
_In_ DWORD wSealedtextSize,
_In_reads_bytes_opt_(dwAuthdataSize) LPCVOID pAuthdata,
_In_ DWORD dwAuthdataSize,
_Out_writes_bytes_to_(dwPlaintextSize) LPCVOID pPlaintext,
_Inout_ DWORD dwPlaintextSize
Key KeyForData,
EnclaveIdentity Identity
)
```

【０１４５】

[0167]ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＵｎｓｅａｌは、抽象化プラットフォーム１９５４内で実施され得、宛先エンクレーブ１９５２からのリクエストに応答して動作し得る。ＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＵｎｓｅａｌは、例えば、しかしＤＳＥ１９３２への宛先エンクレーブ１９５２のアテステーション中に生成される鍵を用いてメッセージを暗号化すること、または宛先エンクレーブの公開鍵を用いて宛先エンクレーブに送信されるメッセージを暗号化することによって、ＤＳＥ１９３２へのセキュア通信チャネルを確立し得る。ＤＳＥは、アテステーションによってなど、リクエスト先の（宛先）エンクレーブのアイデンティティを検証し、リクエストされた密封データを開封し、開封されたデータをリクエスト先のエンクレーブにセキュアに送信し得る。リクエスト先のエンクレーブが複数のアイデンティティを有する実施形態において、特定のアイデンティティは、アイデンティティパラメーター内で特定され得る。複数のエンクレーブデータセ

ットが単一のエンクレーブアイデンティティのために格納される実施形態において、ＫｅｙＦｏｒＤａｔａパラメーターは、データセットが密封されたときにＤｉｓｔｒｉｂｕｔｅｄＥｎｃｌａｖｅＳｅａｌで使用された同じＫｅｙＦｏｒＤａｔａ値を使用することによって、どの密封データセット（特定されたアイデンティティのための）がリクエストされるかを特定し得る。

【０１４６】

[0168]いくつかの実施形態において、データを開封する権限を与えられるエンクレーブのアイデンティティは、標的とする権限を与えられた標的とするエンクレーブの公開鍵によって特定され得る（ＳｅｔＯｆＴａｒｇｅｔＥｎｃｌａｖｅｓパラメーター内など）。この実施形態において、ＤＳＥへの宛先エンクレーブのアテステーションは、必要ではない場合があるが、開封されたデータは、その後、特定された公開鍵のうちの１つを使用して暗号化される際にのみ提供され得る。標的とされたエンクレーブのみが復号するための対応するプライベート鍵を有すると仮定すると、標的とされたエンクレーブのみが、開封されたデータへのアクセスを有することになる。

【０１４７】

[0169]図１９には描かれていない実施形態において、分散密封エンクレーブ（ＤＳＥ）１９３２の機能は、それ自体が、複数のＤＳＥにわたって分散され得る。例えば、ＤＳＥ機能性は、冗長性およびフォールトトレランスのために、複数のコンピューター上の複数のＤＳＥにわたって分散され得る。この例では、任意の複製ＤＳＥは、密封または開封リクエストをサービスすることができる場合がある。密封データ１９３８は、一旦密封／暗号化されると、クラウドストレージサーバーにわたって複製されることなど、どこにでも安全に格納され得るということに留意されたい。

【０１４８】

[0170]分散データ密封は、コンピューター間のエンクレーブ作業負荷の移動を可能にし得る。例えば、ＤＳＥによって密閉される発信元エンクレーブデータは、第１のクラウドサーバー上の発信元エンクレーブの状態データであり得、これは開封後に第２のクラウドサーバー上の宛先エンクレーブ内へロードされ得る。これは、図１８に関して上に説明されるものと同様に行われ得る。セキュア動作は、発信元エンクレーブ内で実行を開始し得る。その後、おそらくは発信元エンクレーブ内での実行が中断された後、発信元エンクレーブの状態は、ＤＳＥへ密封され、次いで、宛先エンクレーブが、発信元エンクレーブにおいて開始されたセキュア動作を継続する準備が整ったときに、宛先エンクレーブへ開封され得る。

【０１４９】

[0171]図２０は、密封エンクレーブまたはＤＳＥによって実施され得るような分散データ密封および開封のための例示的なフローチャートである。ボックス２００２～２００６は分散データ密封に対応する一方、ボックス２００８～２０１０は、分散データ開封に対応する。発信元エンクレーブから生じるエンクレーブデータセットを密封するためのリクエストに応答して、密封エンクレーブ（またはＤＳＥ）は、ボックス２００２において、アテステーションレポートまたはクォートを発信元エンクレーブに送信することによって、発信元エンクレーブに対して自らを証明し得る。発信元エンクレーブは、密封エンクレーブのアテステーションレポート内のアイデンティティ値および署名を調査することによって、密封エンクレーブのアイデンティティを、本物かつ信頼できる密封エンクレーブとして検証し得る。ボックス２００４において、密封エンクレーブは、許可されたリストおよび密封されるべきエンクレーブデータを受信する。これらは、図１９に関して上に説明されるようにセキュアチャネルを介して受信され得る。任意選択のボックス２００６において、密封エンクレーブは、例えば、データが、コンピューターファイルシステム内など密封エンクレーブのセキュアコンテナの外側に格納される場合、発信元エンクレーブのデータを自らに密封し得る。宛先エンクレーブのためのデータを開封するために、宛先エンクレーブは、ボックス２００８において、アテステーションレポートまたはクォートを提供することなどによって、密封エンクレーブに対して自らを証明し得る。ボックス２０１

０において、宛先エンクレーブのアイデンティティは、宛先エンクレーブのアテステーションレポートを調査することなどによって、検証され得る。ボックス２０１２において、密封エンクレーブは、宛先エンクレーブの認証されたアイデンティティがデータと一緒に受信される許可リスト内に含まれることを検証することによって、宛先エンクレーブが発信元エンクレーブからのデータを開封することを許可されているかどうかを決定する。一旦許可が確認されると、エンクレーブデータは、ボックス２０１４において、それが密封された場合には開封され、次いで、セキュアチャネルを介して宛先エンクレーブに送信され得る。

【０１５０】

Ｋｅｙ　Ｖａｕｌｔエンクレーブ

[0172]Ｋｅｙ　Ｖａｕｌｔは、エンクレーブと共に実施され得る。Ｋｅｙ　Ｖａｕｌｔは、データを暗号化し復号するための暗号化システムの鍵などの鍵を、Ｋｅｙ　Ｖａｕｌｔのクライアントのためにセキュアに保持する。Ｋｅｙ　Ｖａｕｌｔは、追加的に、データを暗号化し復号すること、データに署名すること、および既存の鍵から新しい鍵を導出することなど、鍵を用いた動作を実施し得る。Ｋｅｙ　Ｖａｕｌｔは、エンクレーブとして実施されるとき、秘密暗号化鍵の非常にセキュアな記憶装置および秘密暗号化鍵を用いた処理を提供し得る。加えて、Ｋｅｙ　Ｖａｕｌｔエンクレーブのソフトウェアアテステーションは、Ｖａｕｌｔクライアントが本物かつ信頼できるＫｅｙ　Ｖａｕｌｔと通信しているというＶａｕｌｔクライアントへの高いレベルの確証を提供し得る。高度にセキュアなシステムは、Ｖａｕｌｔロックされた鍵を用いてＫｅｙ　Ｖａｕｌｔエンクレーブ上に構築され得、それによりＫｅｙ　Ｖａｕｌｔの内側に格納される鍵は、Ｋｅｙ　Ｖａｕｌｔの外側のいかなるクライアントにも決して解放されず、また、いくつかの場合において、Ｖａｕｌｔロックされた鍵は、Ｋｅｙ　Ｖａｕｌｔエンクレーブの内側に格納される（またはおそらくはそれに密閉される）場合にしか存在することができない。

【０１５１】

[0173]図２１は、例示的なＫｅｙ　Ｖａｕｌｔエンクレーブのブロック図である。エンクレーブ２１２２は、第２のネイティブエンクレーブプラットフォーム２１２６のセキュアエンクレーブコンテナの内側のＫｅｙ　Ｖａｕｌｔである。図２１の例では、Ｋｅｙ　Ｖａｕｌｔエンクレーブ２１２２のクライアント２１１２もエンクレーブであり、第１のネイティブエンクレーブプラットフォーム２１１６のセキュアエンクレーブコンテナの内側でホストされる。エンクレーブ２１１２、２１２２は、それらのそれぞれのネイティブプラットフォーム２１１６、２１２６と、それぞれのエンクレーブ抽象化プラットフォーム２１１４、２１２４を介して相互作用し得る。他の実施形態において、一方または両方の抽象化プラットフォーム２１１４、２１２４は、エンクレーブ２１１２および／または２１２２がネイティブプラットフォーム２１１６、２１２６と直接的に相互作用する場合には存在しない場合がある。

【０１５２】

[0174]Ｋｅｙ　Ｖａｕｌｔエンクレーブ２１２２は、通信チャネル２１５０を介してＶａｕｌｔクライアント２１１２と通信し得る。いくつかの実施形態において、通信チャネル２１１２は、通信チャネル２１５０を介して送信されるメッセージの機密性、完全性、および／または新鮮性の確証を提供するセキュア通信チャネルであり得る。そのようなセキュア通信チャネルの機密性および完全性は、図５および図６にあるように、アテステーションプロセスの部分として生成される共有鍵を使用して、図２および図３にあるように、例えば、暗号化および署名により確立され得る。

【０１５３】

[0175]ソフトウェアアテステーションは、部分的に、他のサイズの通信チャネル上のエンティティのアイデンティティの確証を提供することによってセキュリティを提供する。Ｋｅｙ　Ｖａｕｌｔエンクレーブ２１２２をＶａｕｌｔクライアントに証明することによって、クライアントは、鍵または他のクリアテキストデータなどの秘密をＫｅｙ　Ｖａｕｌｔに送信する前に、エンクレーブ２１２２が自ら語る通りのものであるという確証を得

ることができる。図２１に描写されるように、エンクレーブでもあるクライアントの場合は、逆もまた然りである。Ｖａｕｌｔエンクレーブ２１１２をＫｅｙ Ｖａｕｌｔエンクレーブ２１２２に証明することによって、Ｋｅｙ Ｖａｕｌｔは、鍵または他のクリアテキストデータなどの秘密をクライアントに明らかにする前に、クライアントが自ら語る通りのものであるという確証を得ることができる。

【０１５４】

[0176]Ｖａｕｌｔロックされた鍵および導出された鍵を用いたＫｅｙ Ｖａｕｌｔシステムは、特に暗号化鍵がＶａｕｌｔロックされた鍵から導出される場合、柔軟でありセキュアな状態を変化させるセキュリティシステムを構築するために使用され得る。公開である場合とそうでない場合とがある鍵導出関数は、第１の鍵から複数の鍵を生成するために使用され得る。第１の鍵（ルート秘密）は、最高レベルのセキュリティのためにＶａｕｌｔロックされ得、第１の鍵から導出される鍵は、暗号化目的のために使用され得る。導出鍵が不正アクセスされる場合、第１の鍵を保持するＫｅｙ Ｖａｕｌｔにアクセスするまたはそれを変化させる必要なく、新しい導出鍵が既存のシステム内で生成され得る。

【０１５５】

[0177]例示的なＫｅｙ Ｖａｕｌｔエンクレーブ（ＫＶＥ）は、エンクレーブを使用して鍵ストレージ、生成、導出、分散、暗号化、復号、および署名を提供するクラウドベースのＫｅｙ Ｖａｕｌｔシステムである。ＫＶＥは、その正確なハッシュ（そのセキュアコンテナのコンテンツのハッシュ）によって、またはその作成者によって指定されるかもしくはそれと関連付けられる任意の識別子によって、識別され得る。後者の場合、エンクレーブは、識別子の衝突に起因するクラッシュおよびセキュリティ違反を避けるために、その作成者のプライベート鍵を用いて署名され得る。

【０１５６】

[0178]Ｋｅｙ Ｖａｕｌｔクライアントは、以下の例示的なプリミティブを使用してＫｅｙ Ｖａｕｌｔシステムと相互作用し得る。例示的なＳｔｏｒｅＫｅｙ関数プロトタイプは、ＳｔｏｒｅＫｅｙ（［ｉｎ］Ｋｅｙｎａｍｅ，［ｉｎ］ＫｅｙＴｙｐｅ，［ｉｎ］ＫｅｙＶａｌｕｅ，［ｉｎ］Ｐｏｌｉｃｙ）である。ＳｔｏｒｅＫｅｙは、Ｋｅｙ Ｖａｕｌｔ内に所与の鍵を格納し、それを所与の名前と関連付ける。鍵タイプはまた、鍵と関連付けられ、その鍵を用いて行われ得ることを制限する。例えば、いくつかの鍵は、暗号化のためだけに使用されるべきであり、他のものは署名などのために使用されるべきである。また、特定の鍵は、特定の暗号アルゴリズムとのみ使用され得る。ポリシーは、鍵の使用をさらに制限するためにポリシーを特定し得る。例えば、ポリシーは、鍵を取得するおよび／または鍵を使用することが許可されるエンクレーブアイデンティティのセットを特定し得る。ポリシーはまた、例えば、鍵が特定の日時に破壊されるべきである、または鍵を使用した動作の速度が制限されるべきであるという時間的特性を特定し得る。

【０１５７】

[0179]例示的なＧｅｎｅｒａｔｅＫｅｙ関数プロトタイプは、ＧｅｎｅｒａｔｅＫｅｙ（［ｉｎ］ｋｅｙＮａｍｅ，［ｉｎ］ｋｅｙＴｙｐｅ，［ｉｎ］Ｐｏｌｉｃｙ）である。

[0180]ＧｅｎｅｒａｔｅＫｅｙは、特定のタイプの新しい鍵を生成し、それをＫｅｙ Ｖａｕｌｔの内側に格納されたままにする、すなわち、鍵はＫｅｙ Ｖａｕｌｔを決して離れない。

【０１５８】

[0181]例示的なＧｅｔＫｅｙ関数プロトタイプは、ＧｅｔＫｅｙ（［ｉｎ］ＫｅｙＮａｍｅ，［ｏｕｔ］ＫｅｙＶａｌｕｅ）である。

【０１５９】

[0182]ＧｅｔＫｅｙは、Ｋｅｙ Ｖａｕｌｔの内側に格納される鍵をフェッチする。これらのプリミティブは、典型的には、セキュア通信チャネルを介して実施され、プリミティブを呼び出すコードは、典型的には、信頼できる環境内で実行する。そのようなコンテキストにおいて、多くの場合、Ｋｅｙ Ｖａｕｌｔから鍵を取得することは認められている。

【０１６０】
[0183]例示的なＤｅｌｅｔｅＫｅｙ関数プロトタイプは、ＤｅｌｅｔｅＫｅｙ（［ｉｎ］ｋｅｙＮａｍｅ）である。
[0184]ＤｅｌｅｔｅＫｅｙは、Ｋｅｙ　Ｖａｕｌｔから鍵を消去する。
【０１６１】
[0185]例示的なＤｅｒｉｖｅＫｅｙ関数プロトタイプは、ＤｅｒｉｖｅＫｅｙ（［ｉｎ］ｎｅｗＫｅｙＮａｍｅ，［ｉｎ］ＫｅｙＮａｍｅ，［ｉｎ］ｋｄｆＩｄｅｎｔｉｆｉｅｒ，［ｉｎ］ＡｄｄｉｔｉｏｎａｌＤａｔａ）である。
【０１６２】
[0186]ＤｅｒｉｖｅＫｅｙは、ｋｅｙＮａｍｅによって識別される鍵およびＡｄｄｉｔｉｏｎａｌＤａｔａ内で引き渡されるデータに基づいて新しい鍵を導出するために、ｋｄｆＩｄｅｎｔｉｆｉｅｒによって識別される暗号化鍵導出関数（ＫＤＦ）を使用する。
【０１６３】
[0187]例示的なＥｎｃｒｙｐｔ関数プロトタイプは、Ｅｎｃｒｙｐｔ（［ｉｎ］ＫｅｙＮａｍｅ，［ｉｎ］ａｌｇｏｒｉｔｈｍ，［ｉｎ］ｄａｔａ，［ｏｕｔ］ｅｎｃｒｙｐｔｅｄＤａｔａ）である。
【０１６４】
[0188]Ｅｎｃｒｙｐｔは、リクエストされたアルゴリズムを使用して、ＫｅｙＮａｍｅによって識別される鍵を用いてデータを暗号化する。
[0189]例示的なＤｅｃｒｙｐｔ関数プロトタイプは、Ｄｅｃｒｙｐｔ（［ｉｎ］ＫｅｙＮａｍｅ，［ｉｎ］ａｌｇｏｒｉｔｈｍ，［ｉｎ］ｅｎｃｒｙｔｅｄＤａｔａ，［ｏｕｔ］ｄａｔａ）である。
【０１６５】
[0190]Ｄｅｃｒｙｐｔは、リクエストされたアルゴリズムを使用して、ＫｅｙＮａｍｅによって識別される鍵を用いてデータを復号する。
[0191]例示的なＳｉｇｎ関数プロトタイプは、Ｓｉｇｎ（［ｉｎ］ＫｅｙＮａｍｅ，［ｉｎ］ａｌｇｏｒｉｔｈｍ，［ｉｎ］ｄａｔａ，［ｏｕｔ］ｓｉｇｎａｔｕｒｅ）である。
【０１６６】
[0192]Ｓｉｇｎは、リクエストされたアルゴリズムを使用して、ＫｅｙＮａｍｅによって識別される鍵を用いてデータに署名する。
[0193]例示的なＶｅｒｉｆｙＳｉｇｎａｔｕｒｅ関数プロトタイプは、ＶｅｒｉｆｙＳｉｇｎａｔｕｒｅ（［ｉｎ］ＫｅｙＮａｍｅ，［ｉｎ］ａｌｇｏｒｉｔｈｍ，［ｉｎ］ｓｉｇｎａｔｕｒｅ，［ｏｕｔ｝ｂｏｏｌ　ｓｉｇｎａｔｕｒｅＩｓＣｏｒｒｅｃｔ）である。
【０１６７】
[0194]ＶｅｒｉｆｙＳｉｇｎａｔｕｒｅは、リクエストされたアルゴリズムを使用して、ＫｅｙＮａｍｅによって識別される鍵を用いて署名を検証する。
[0195]上のＫｅｙ　Ｖａｕｌｔプリミティブのすべては、ＫＶＥによりセキュアチャネルを確立することによって実施され得る。チャネルは、図５および図６に関して上に説明されるように、アテステーションを使用し、ディフィー・ヘルマン鍵交換を実施して確立され得る。通信チャネルが確立された後、リクエストは、チャネルを介してセキュアに送信され、応答は、チャネルから読み込まれ得る。チャネルは、交換されるデータの機密性および完全性の保証を提供し得る。
【０１６８】
[0196]別の実施形態において、最初にＫＶＥが実行するとき、ＫＶＥは、公開／プライベート鍵ペアを生成し、またＫＶＥは、公開鍵のためのクォートを生成する。次いで、ＫＶＥは、クォートおよび公開鍵を書き出す一方で、プライベート鍵をエンクレーブの内側に保つ。公開鍵およびクォートは、次いで、Ｋｅｙ　Ｖａｕｌｔを使用することを望むすべてのシステム／コードへ分散され得る。この場合、上のプリミティブの実装形態は、ク

ォートが本物のＫＶＥとやりとりしていることを確かめるためにクォートを検証し、次いで、ＫＶＥの公開鍵を使用してリクエストを暗号化する。リクエストの部分として、プリミティブの実装形態は、ＫＶＥから送信される結果を暗号化し完全性保護するための鍵を含み得る。この実施形態は、アテステーションなしでセキュアな双方向通信チャネルを提供し得る。

【０１６９】

[0197]図２２は、いくつかのＫｅｙ　Ｖａｕｌｔエンクレーブ動作の例示的なフローチャートである。プロセス２２００は、ボックス２２０２において、Ｋｅｙ　Ｖａｕｌｔエンクレーブ内で、暗号化システムにおいて使用される鍵をセキュアに格納することにより開始する。鍵は、例えば、データを暗号化もしくは復号し、暗号化署名を生成するために使用され得るか、または他の鍵を導出するためのルート鍵としてのみ使用され得る。鍵は、例えば、エンクレーブのセキュアコンテナのメモリ空間内に鍵を格納することによって、Ｋｅｙ　Ｖａｕｌｔエンクレーブにセキュアに格納され得る。他の実施形態において、鍵は、鍵データをＫｅｙ　Ｖａｕｌｔエンクレーブへ密封することによってセキュアエンクレーブコンテナの外側にセキュアに保たれ得るか、または、図１９および図２０に関して説明されるように、分散密封エンクレーブによりリモートで密封することによってセキュアに保たれ得る。

【０１７０】

[0198]ボックス２２０４において、Ｋｅｙ　Ｖａｕｌｔエンクレーブは、Ｖａｕｌｔクライアントに対してＶａｕｌｔエンクレーブのアイデンティティを証明するためのアテステーションプロセスを実施する。これは、Ｋｅｙ　Ｖａｕｌｔがなりすましではなく、暗号化されるべき鍵またはデータなどの秘密により信頼され得るという確証をクライアントに与え得る。Ｋｅｙ　Ｖａｕｌｔエンクレーブのアテステーションは、Ｖａｕｌｔクライアントに、Ｋｅｙ　Ｖａｕｌｔエンクレーブのアテステーションレポートまたはアテステーションクォートを送信することを含み得る。Ｋｅｙ　Ｖａｕｌｔクライアントは、次いで、Ｋｅｙ　Ｖａｕｌｔエンクレーブのネイティブエンクレーブプラットフォームと関連付けられた公開鍵を用いてアテステーションレポート内の署名を検証することによって、アテステーションレポートの完全性を検証することができる。例えば、Ｋｅｙ　Ｖａｕｌｔ２１２２のアテステーションレポートは、第２のネイティブプラットフォーム２１２６によって生成され得、Ｖａｕｌｔクライアント２１１２は、第２のネイティブプラットフォーム２１２６と関連付けられた公開鍵を使用してレポート内の署名を検証し得る。このアテステーションプロセスはまた、例えば、図５および図６に示されるように、Ｖａｕｌｔクライアントとの間のセキュア通信チャネルのために使用される鍵を生成し得る。アテステーションレポートは、例えば図１４および図１５に関して上に説明されるような様々なやり方で決定され得るＫｅｙ　Ｖａｕｌｔエンクレーブのアイデンティティを含み得る。アイデンティティは、例えば、Ｋｅｙ　Ｖａｕｌｔエンクレーブのセキュアコンテナの全コンテンツのハッシュ、Ｋｅｙ　Ｖａｕｌｔエンクレーブのオーサー／作成者によって指定される固有識別子のみのハッシュ、またはコンテナのコンテンツの一部分および固有識別子の組合せのハッシュに基づき得る。

【０１７１】

[0199]いくつかのＫｅｙ　Ｖａｕｌｔエンクレーブ動作はまた、Ｖａｕｌｔクライアントのアイデンティティの確証を必要とし得る。例えば、データを復号することまたは鍵を漏らすこと（ＤｅｃｒｙｐｔまたはＧｅｔＫｅｙプリミティブなどにより）は、そのような確証を必要とし得る。これらの状況において、Ｖａｕｌｔクライアントもエンクレーブである場合、任意選択のボックス２２０８は、Ｋｅｙ　Ｖａｕｌｔエンクレーブによって、Ｖａｕｌｔクライアントのアイデンティティを検証するためのアテステーションプロセスを含む。ボックス２２０８のアテステーションプロセスは、Ｋｅｙ　Ｖａｕｌｔエンクレーブにおいて、Ｖａｕｌｔクライアントのアテステーションレポートまたはクォートを受信することを含み得る。

【０１７２】

[0200]任意選択のボックス２２１０において、セキュア通信チャネルが、Ｋｅｙ　ＶａｕｌｔとＫｅｙ　Ｖａｕｌｔエンクレーブとの間で確立され得る。セキュアな通信は、Ｖａｕｌｔクライアントと Ｋｅｙ　Ｖａｕｌｔエンクレーブとの間で、暗号化されるべき鍵またはデータなどの秘密を引き渡すことが必要とされ得る。ボックス２００４または２００８のアテステーションプロセスは、例えば、図５および図６に示されるように、Ｖａｕｌｔクライアントと Ｋｅｙ　Ｖａｕｌｔエンクレーブとの間にセキュア通信チャネルを作成するために使用され得る鍵を生成し得る。代替的に、メッセージの宛先の任意の知られている公開鍵が、メッセージをセキュアに送信するために使用され得る。

【０１７３】

[0201]ボックス２２１２において、上に説明されるＫｅｙ　Ｖａｕｌｔプリミティブのうちの１つなどの鍵動作が、Ｋｅｙ　Ｖａｕｌｔエンクレーブの内側で実施され得る。この動作中、鍵データは、Ｋｅｙ　Ｖａｕｌｔエンクレーブのセキュアコンテナのアドレス空間にのみ格納され得る。例示的なプリミティブとしては、ＤｅｒｉｖｅＫｅｙ、Ｄｅｃｒｙｐｔ、Ｓｉｇｎなどが挙げられる。

【０１７４】

[0202]プロセス２２００は、Ｋｅｙ　Ｖａｕｌｔエンクレーブが鍵を既に知っていると仮定する。ＳｔｏｒｅＫｅｙまたは ＧｅｎｅｒａｔｅＫｅｙなどのいくつかのＫｅｙ　Ｖａｕｌｔエンクレーブ動作またはプリミティブの場合、動作の順序は、プロセス２２００において描写されるものとは異なり得ることに留意されたい。例えば、ＧｅｎｅｒａｔｅＫｅｙの場合、鍵生成動作（ボックス２２１２にあるような）は、ボックス２２０２の安全な格納動作の前に発生することになる。そのような動作順序は、図２３のボックス２３０２～２３０８に描写される。

【０１７５】

[0203]図２３は、Ｖａｕｌｔロックされた鍵を用いたＫｅｙ　Ｖａｕｌｔエンクレーブを作成および使用するための例示的なフローチャートである。プロセス２３００のボックス２３０２～２３０８において、新しい鍵がＫｅｙ　Ｖａｕｌｔエンクレーブ内で導出される。ボックス２３１０～２３１６において、新しく導出された鍵は、復号動作を実施するために使用される。これは、Ｖａｕｌｔロックされた鍵の例示的な使用であり、それにより、すべての鍵動作は、Ｋｅｙ　Ｖａｕｌｔエンクレーブを用いて実施され、鍵は、Ｖａｕｌｔクライアントに決して提供されない。さらに、この例における新しい鍵は、それがＫｅｙ　Ｖａｕｌｔエンクレーブ自体の中で作成（導出）されたために、Ｋｅｙ　Ｖａｕｌｔエンクレーブの外側には決して存在することができず、また、Ｖａｕｌｔクライアントまたは他の場所からＫｅｙ　Ｖａｕｌｔエンクレーブに決して提供されない。いくつかの実施形態および鍵使用ポリシーでは、Ｖａｕｌｔロックされた鍵は、Ｋｅｙ　Ｖａｕｌｔエンクレーブへ鍵を密封した後にさえ、Ｋｅｙ　Ｖａｕｌｔエンクレーブのセキュアコンテナを決して離れないという点で一過性であり得る。通信チャネルを一時的にセキュアにするために使用される導出鍵と一緒に発生し得るなど、そのような一過性の鍵は、Ｋｅｙ　Ｖａｕｌｔエンクレーブのコンテナが破壊または終了されるときにはどこにも存在しなくなり得る。図２３のプロセスは、Ｖａｕｌｔロックされた鍵がどのように使用され得るかを例証する一方、図２３のプロセスは、例えば、鍵使用ポリシーが、鍵がその作成をリクエストしたクライアントへ戻ることを許可した場合、Ｖａｕｌｔロックされない鍵と共に使用され得る。

【０１７６】

[0204]ボックス２３０２において、Ｋｅｙ　Ｖａｕｌｔエンクレーブは、Ｖａｕｌｔクライアントに対して自らを証明する。これは、クライアントがボックス２３１２において暗号化されるべき秘密を提供することから、クライアントによって要求され得る。ボックス２３０４において、Ｋｅｙ　Ｖａｕｌｔエンクレーブは、例えば、Ｖａｕｌｔクライアントから、鍵使用ポリシーの標示を受信し得る。標示は、例えば、ポリシーを特定するデータ構造であり得るか、または、鍵使用ポリシーのレジストリーと共に使用されるべき識別子であり得る。鍵使用ポリシー自体は、この鍵がいかなるＶａｕｌｔクライアントにも

提供されるべきではないことを示し得る。ボックス２３０６において、新しい鍵が、以前に知られているルート鍵から、例えば、上に説明されるＤｅｒｉｖｅＫｅｙプリミティブにより、導出される。新しい鍵を導出するためのリクエスト（描写されない）は、例えば、Ｖａｕｌｔクライアントから、Ｋｅｙ　Ｖａｕｌｔエンクレーブによって受信され得る。ボックス２３０８において、新しく導出された鍵は、受信した鍵使用ポリシーに従ってセキュアに格納され得る。

【０１７７】

[0205]Ｖａｕｌｔクライアントは、ボックス２３１０において、Ｋｅｙ　Ｖａｕｌｔエンクレーブに対して自らを証明し得る。アテステーションプロセスは、Ｋｅｙ　Ｖａｕｌｔエンクレーブにおいて、Ｖａｕｌｔクライアントのアテステーションレポートまたはクォートを受信することを含み得る。受信した鍵使用ポリシーは、新しい鍵の一部またはすべての使用を、ソフトウェアアテステーションにより認証されるリクエスト元からのリクエストに制限し得る。ボックス２３１２～２３１６において、上のＤｅｃｒｙｐｔプリミティブなどの復号動作は、ボックス２３０６において、導出される鍵を使用して実施される。他の実施形態において、暗号化、署名、署名を検証すること、およびボックス２３０６において導出される鍵から別の新しい鍵を導出すること（ルート鍵から第２の生成鍵を導出すること）など、他の動作が、Ｖａｕｌｔロックされた鍵を用いて実施され得る。ボックス２３１２において、暗号化されたデータバッファが、Ｖａｕｌｔクライアントから受信される。受信した暗号化データは、ボックス１３１４において、導出鍵を用いて復号され、結果として生じる復号されたデータ（復号されたデータバッファ内の）は、ボックス２３１６において、セキュア通信チャネルを介してＶａｕｌｔクライアントに送信される。

【０１７８】

[0206]一実施形態において、エンクレーブ識別方法は、プロセッサーおよびメモリを備えるコンピューティングデバイスによって実施され、本方法は、

[0207]アイデンティティタイプ、およびインスタンス化されたエンクレーブに関連した動作のリクエストを受信するステップと、

[0208]アイデンティティタイプ、およびエンクレーブがインスタンス化されたエンクレーブ画像から導出される情報に基づいて、エンクレーブのアイデンティティ値を決定するステップと、

[0209]アイデンティティ値を用いて動作を実施するステップと、を含む。

【０１７９】

[0210]一実施形態において、本方法は、

[0211]エンクレーブ画像のオーサーに関連した公開鍵を用いてエンクレーブ画像内の署名を検証することによって、アイデンティティ値の完全性を検証するステップをさらに含む。

【０１８０】

[0212]一実施形態において、アイデンティティ値は、ハッシュ関数の出力として決定され、また

[0213]アイデンティティタイプに少なくとも部分的に基づいて、エンクレーブ画像のアイデンティティ部を決定するステップと、

[0214]アイデンティティ部についてハッシュ関数をコンピューティングするステップと、をさらに含む。

【０１８１】

[0215]一実施形態において、エンクレーブ画像は、追加の依存エンクレーブ画像への参照を含み、また

[0216]アイデンティティタイプに少なくとも部分的に基づいて、追加の依存エンクレーブ画像のうちのどれがハッシュ関数への入力として含まれるかを決定するステップをさらに含む。

【０１８２】

[0217]一実施形態において、アイデンティティ部は、インスタンス化されたエンクレーブのインスタンス化中にセキュアエンクレーブコンテナ内へコピーされるバイナリコード、および実行可能コードではない１つまたは複数の識別子、のうちの１つまたは複数を含む。

【０１８３】

[0218]一実施形態において、

[0219]アイデンティティ値は、ハッシュ関数の出力として決定され、

[0220]エンクレーブは、複数のエンクレーブ画像を用いてインスタンス化されており、

[0221]アイデンティティタイプは、複数のエンクレーブ画像のうちのどれが、アイデンティティ値を決定するためにハッシュ関数に少なくとも部分的に含まれるかを示す。

【０１８４】

[0222]一実施形態において、

[0223]動作は、アイデンティティ値を有するアテステーションレポートを生成することを含む。

【０１８５】

[0224]実施形態において、

[0225]動作は、アイデンティティ値を有するデータを密封することによって、データをエンクレーブへ密封することを含む。

【０１８６】

[0226]一実施形態において、動作は、単調カウンターを増分することを含み、単調カウンターは、アイデンティティ値によって識別される。

[0227]一実施形態において、動作は、信頼できる時刻測定を行うことを含み、信頼できる時刻は、アイデンティティ値に基づいて決定される。

【０１８７】

[0228]一実施形態において、システムは、少なくともプロセッサーと命令を格納するメモリとを備え、命令は、システムによって実行されるとき、少なくとも

[0229]アイデンティティタイプ、およびインスタンス化されたエンクレーブに関連した動作のリクエストを受信することと、

[0230]アイデンティティタイプ、およびエンクレーブがインスタンス化されたエンクレーブ画像から導出される情報に基づいて、エンクレーブのアイデンティティ値を決定することと、

[0231]アイデンティティ値を用いて動作を実施することと、を引き起こす。

【０１８８】

[0232]一実施形態において、命令は、少なくとも

[0233]エンクレーブ画像のオーサーに関連した公開鍵を用いてエンクレーブ画像内の署名を検証することによって、アイデンティティ値の完全性を検証することをさらに引き起こす。

【０１８９】

[0234]一実施形態において、アイデンティティ値は、ハッシュ関数の出力として決定され、また命令は、少なくとも

[0235]アイデンティティタイプに少なくとも部分的に基づいて、エンクレーブ画像のアイデンティティ部を決定することと、

[0236]アイデンティティ部についてハッシュ関数をコンピューティングすることとをさらに引き起こす。

【０１９０】

[0237]一実施形態において、エンクレーブ画像は、追加の依存エンクレーブ画像への参照を含み、また命令は、少なくとも

[0238]アイデンティティタイプに少なくとも部分的に基づいて、追加の依存エンクレーブ画像のうちのどれがハッシュ関数への入力として含まれるかを決定することをさらに引き起こす。

【０１９１】
[0239]一実施形態において、アイデンティティ部は、インスタンス化されたエンクレーブのインスタンス化中にセキュアエンクレーブコンテナ内へコピーされるバイナリコード、および実行可能コードではない１つまたは複数の識別子、のうちの１つまたは複数を含む。
【０１９２】
[0240]一実施形態において、アイデンティティ値は、ハッシュ関数の出力として決定され、
[0241]エンクレーブは、複数のエンクレーブ画像を用いてインスタンス化されており、
[0242]アイデンティティタイプは、複数のエンクレーブ画像のうちのどれが、アイデンティティ値を決定するためにハッシュ関数に少なくとも部分的に含まれるかを示す。
【０１９３】
[0243]一実施形態において、動作は、アイデンティティ値を有するアテステーションレポートを生成することを含む。
[0244]一実施形態において、動作は、アイデンティティ値を有するデータを密封することによって、データをエンクレーブへ密封することを含む。
【０１９４】
[0245]一実施形態において、動作は、単調カウンターを増分することを含み、単調カウンターは、アイデンティティ値によって識別される。
[0246]一実施形態において、動作は、信頼できる時刻測定を行うことを含み、信頼できる時刻は、アイデンティティ値に基づいて決定される。
【０１９５】
[0247]一実施形態において、エンクレーブ識別方法は、
[0248]アイデンティティタイプ、およびインスタンス化されたエンクレーブに関連した動作のリクエストを受信するステップと、
[0249]アイデンティティタイプ、およびエンクレーブのコンテナに格納されるデータに基づいて、エンクレーブのアイデンティティ値を決定するステップと、
[0250]アイデンティティ値を用いて動作を実施するステップと、を含む。
【０１９６】
[0251]請求項２１に記載のエンクレーブ識別方法であって、
[0252]エンクレーブのコンテナに格納されるデータは、複数のアイデンティティ値を含み、
[0253]アイデンティティ値は、アイデンティティタイプに基づいて複数のアイデンティティ値の中から選択することによって決定される、方法。
【０１９７】
[0254]先のセクションに説明されるプロセス、方法、およびアルゴリズムの各々は、１つまたは複数のコンピューターまたはコンピュータープロセッサーによって実行されるコードモジュールに埋め込まれ得、またこれらによって完全にまたは部分的に自動化され得る。コードモジュールは、ハードドライブ、固体メモリ、光学ディスク、および／または同様のものなどの任意のタイプの非一時的なコンピューター可読媒体またはコンピューター記憶デバイス上に格納され得る。プロセスおよびアルゴリズムは、特定用途向け回路内で部分的または全体的に実施され得る。開示されたプロセスおよびプロセスステップの結果は、揮発性または不揮発性記憶装置などの任意のタイプの非一時的なコンピューター記憶装置に格納され得る。上に説明される様々な特徴およびプロセスは、互いとは無関係に使用され得るか、または様々なやり方で組み合わせられ得る。すべての潜在的な組合せおよび部分的組合せは、本開示の範囲内に入ることが意図される。加えて、特定の方法またはプロセスブロックは、いくつかの実装形態においては省略され得る。本明細書に説明される方法およびプロセスはまた、いかなる特定の順序にも限定されず、それに関連するブロックまたは状態は、適切である他の順序で実施され得る。例えば、説明されたブロックまたは状態は、具体的に開示されるもの以外の順番で実施され得るか、複数のブロックま

たは状態が、単一のブロックまたは状態において組み合わされ得る。例示的なブロックまたは状態は、連続して、並行して、または何らかの他の様式で実施され得る。ブロックまたは状態は、開示された例示的な実施形態に追加され得るか、またはそこから除去され得る。本明細書に説明される例示的なシステムおよびコンポーネントは、説明されるものとは異なって構成され得る。例えば、開示された例示的な実施形態と比較して、要素が追加され得る、除去され得る、または再配置され得る。

【０１９８】

[0255]様々なアイテムは、使用されている間メモリ内または記憶装置上に格納されるものとして例証されること、ならびに、これらのアイテムまたはそれらの部分は、メモリ管理およびデータ完全性の目的のためにメモリと他の記憶デバイスとの間で転送され得るということも理解されたい。代替的に、他の実施形態において、ソフトウェアモジュールおよび／またはシステムの一部またはすべては、メモリ内または別のデバイス上で実行し得、コンピューター間通信を介して、例証されたコンピューティングシステムと通信し得る。さらには、いくつかの実施形態において、システムおよび／またはモジュールの一部またはすべては、１つまたは複数の特定用途向け集積回路（ＡＳＩＣ）、標準集積回路、コントローラ（例えば適切な命令を実行することにより、またマイクロコントローラおよび／または埋め込み式コントローラを含む）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、複合プログラム可能論理デバイス（ＣＰＬＤ）などを含むがこれらに限定されないファームウェアおよび／またはハードウェア内で少なくとも部分的になど、他のやり方で実施または提供され得る。モジュール、システム、およびデータ構造の一部またはすべてはまた、適切なドライブによって、または適切な接続を介して読み込まれるべきハードディスク、メモリ、ネットワーク、またはポータブルメディア物品などのコンピューター可読媒体上に（例えば、ソフトウェア命令または構造化データとして）格納され得る。本明細書および請求項の目的のため、「コンピューター可読記憶媒体」という用語およびそのバリエーションは、波、信号、ならびに／または他の一時的および／もしくは無形の通信媒体を含まない。システム、モジュール、およびデータ構造はまた、ワイヤレスベースおよび有線／ケーブルベースの媒体を含む様々なコンピューター可読伝送媒体上で、生成されたデータ信号として（例えば、搬送波または他のアナログもしくはデジタル伝播信号の部分として）伝送され得、様々な形態（例えば、単一もしくは多重化アナログ信号の部分として、または複数の離散デジタルパケットもしくはフレームとして）をとり得る。そのようなコンピュータープログラム製品はまた、他の実施形態においては他の形態をとり得る。したがって、本開示は、他のコンピューターシステム構成で実践され得る。

【０１９９】

[0256]本明細書中で使用される条件的言語、とりわけ「ｃａｎ」、「ｃｏｕｌｄ」、「ｍｉｇｈｔ」、「ｍａｙ」、「ｅ．ｇ」、および同様のものなどは、別途具体的に記述されない限り、または使用の際に文脈内でそれ以外に理解されない限り、一般に、特定の特徴、要素、および／またはステップを、特定の実施形態が含み、一方で他の実施形態が含まないことを伝えることが意図される。したがって、そのような条件的言語は、一般に、特徴、要素、および／またはステップが、１つもしくは複数の実施形態にいずれの形であれ必要とされること、あるいは１つもしくは複数の実施形態が、オーサー入力もしくはプロンティングを用いて、または用いずに、これらの特徴、要素、および／もしくはステップが含まれるか、または任意の特定の実施形態において実施されるべきであるかを決定するための論理を必ず含むことを示唆することは意図されない。用語「ｃｏｍｐｒｉｓｉｎｇ（備える）」、「ｉｎｃｌｕｄｉｎｇ（含む）」、「ｈａｖｉｎｇ（有する）」、および同様のものは、同義であり、オープンエンド式に包含的に使用され、追加の要素、特徴、行為、動作などを除外しない。また、用語「ｏｒ（または）」は、例えば、要素のリストをつなぐために使用されるとき、用語「ｏｒ（または」がリスト内の要素の１つ、いくつか、またはすべてを意味するように、その包含的な意味で（排他的な意味ではなく）使用される。

【０２００】

[0257]特定の例示的な実施形態が説明されているが、これらの実施形態は、例としてのみ提示されており、本明細書に開示される発明の範囲を限定することは意図されない。したがって、前述の説明のいずれも、任意の特定の特徴、特性、ステップ、モジュール、またはブロックが必要である、または不可欠であることを示唆することは意図されない。実際、本明細書に説明される新規の方法およびシステムは、様々な他の形態で具現化され得、さらには、本明細書に説明される方法およびシステムの形態における様々な省略、置換、および変更は、本明細書に開示される発明の趣旨から逸脱することなくなされ得る。添付の特許請求の範囲およびそれらの均等物は、本明細書に開示される発明の特定の範囲および趣旨に入るような形態または修正を網羅することが意図される。

【図１】

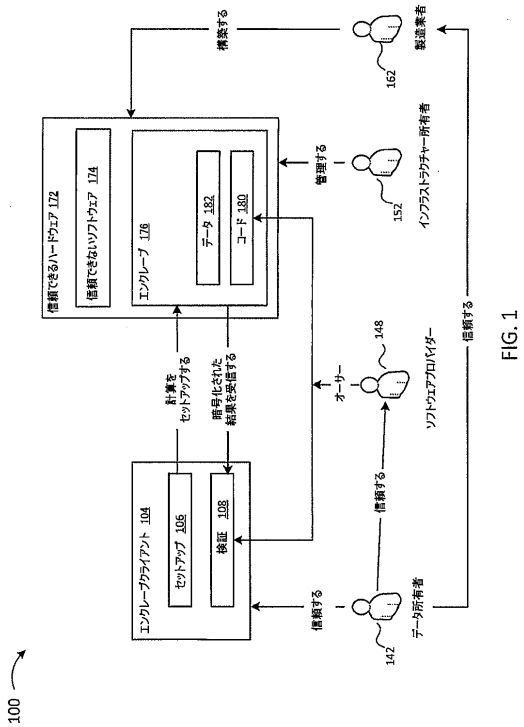


FIG. 1

【図２】

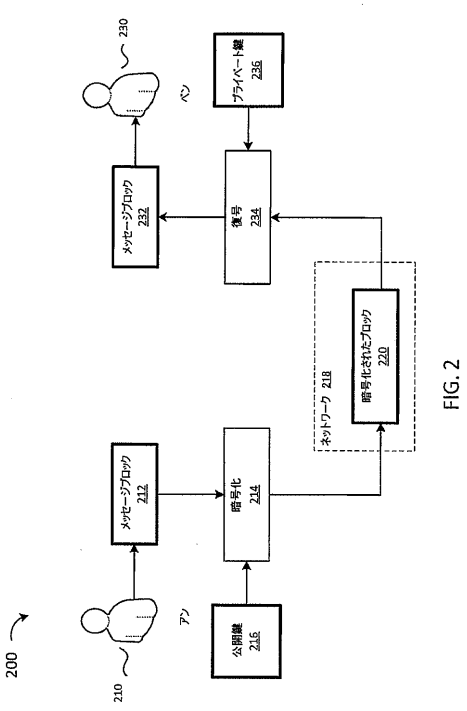


FIG. 2

【図３】

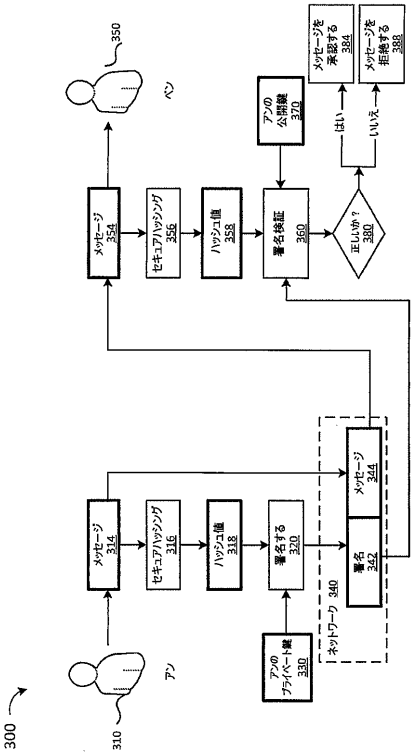


FIG. 3

【図４】

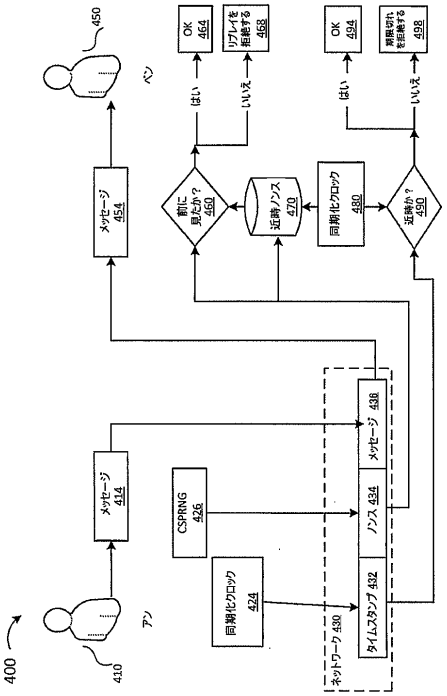


FIG. 4

【図５】

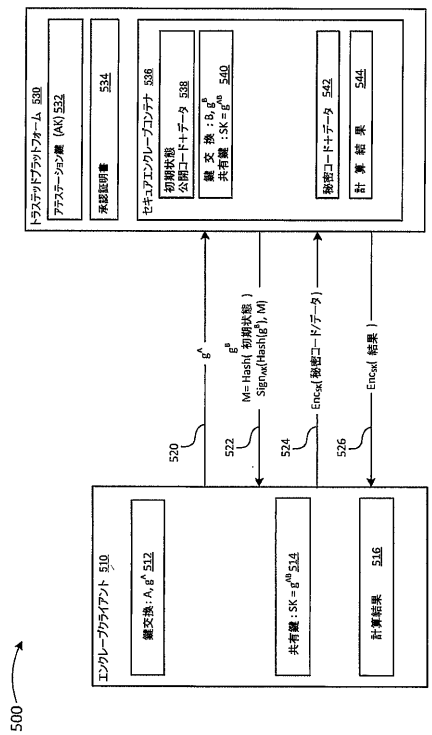


FIG. 5

【図６】

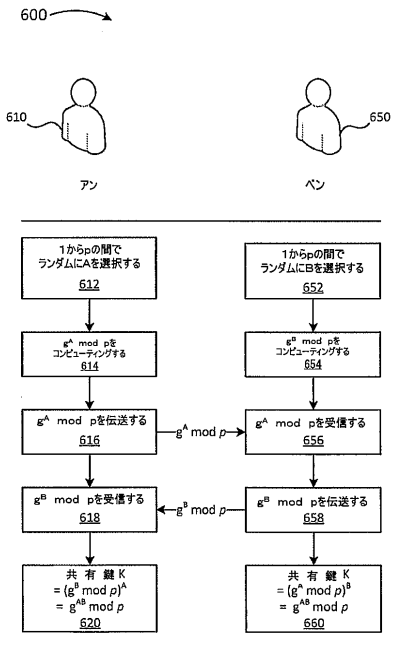


FIG. 6

【図７】

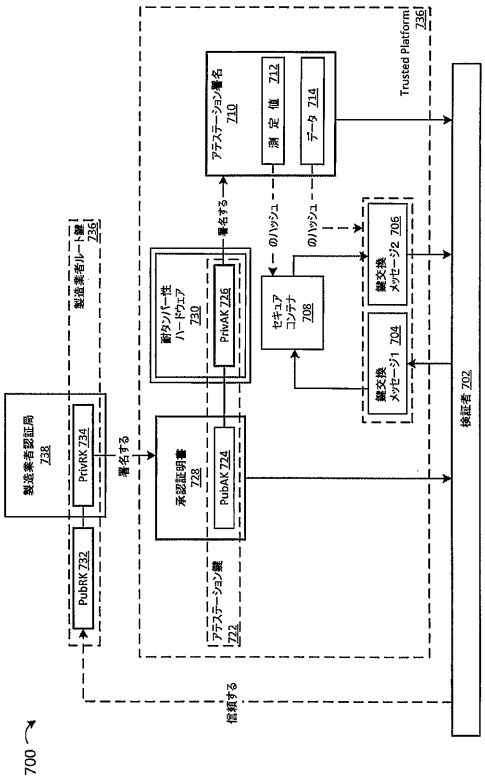


FIG. 7

【図８】

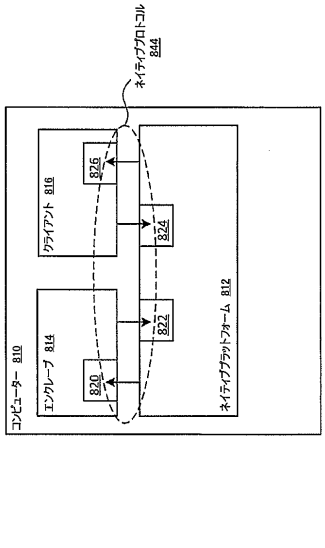


FIG. 8

【図９】

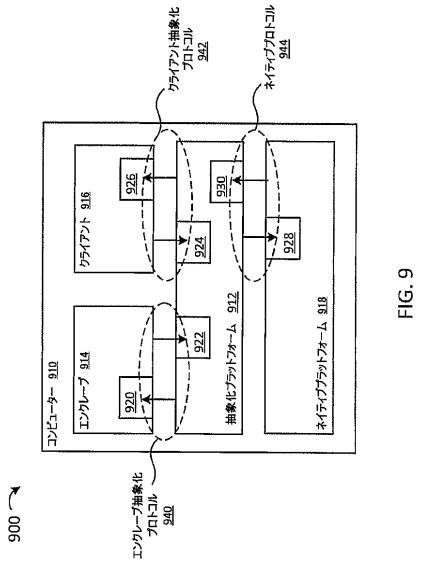


FIG. 9

【図１０】

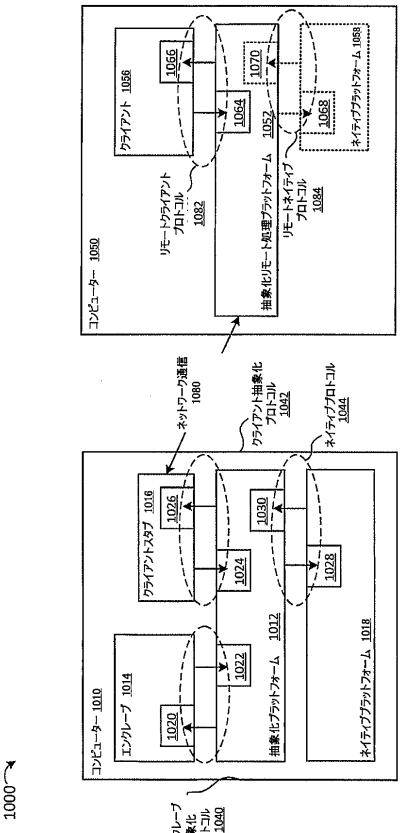


FIG. 10

【図１１】

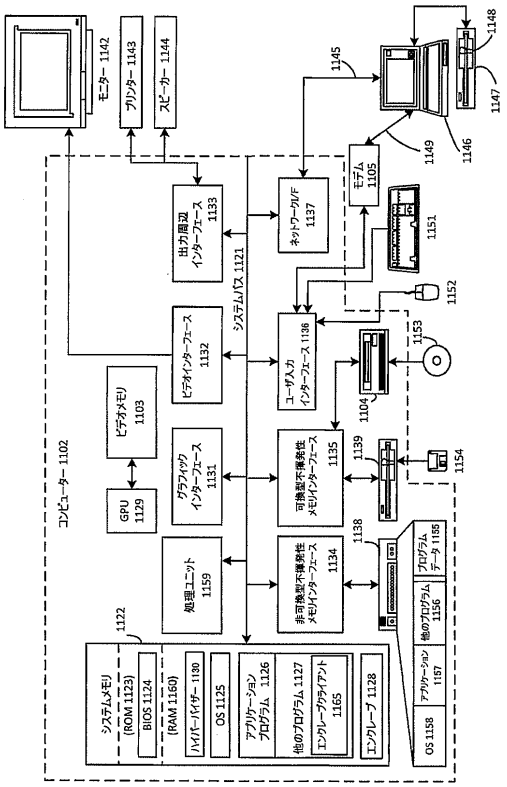


FIG. 11

【図１２】

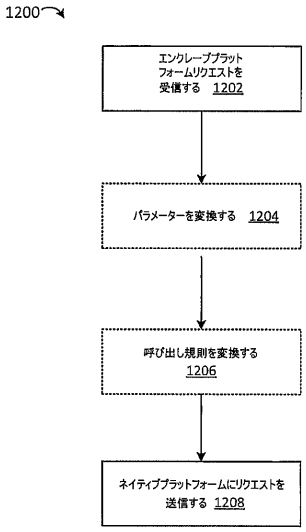


FIG. 12

【図１３】

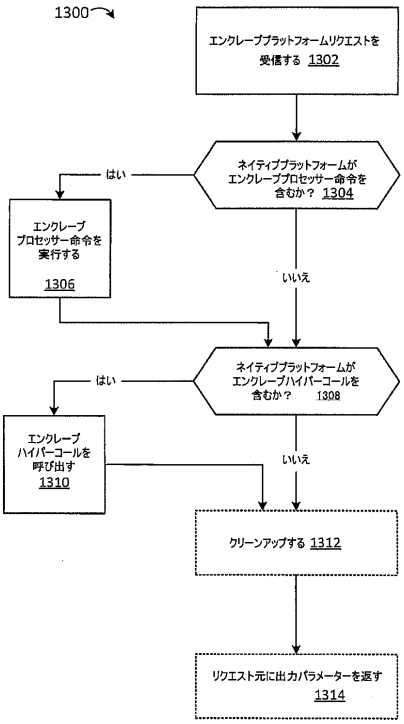


FIG. 13

【図１４】

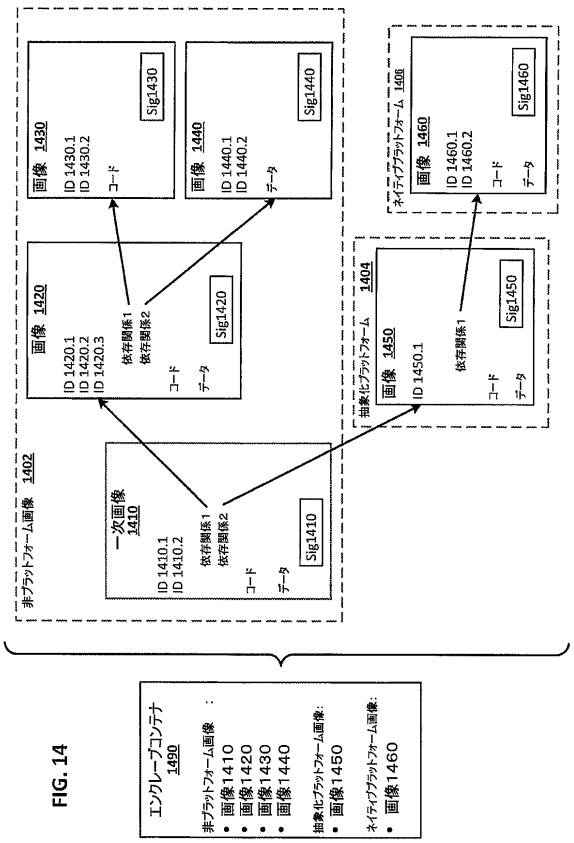


FIG. 14

【図１５】

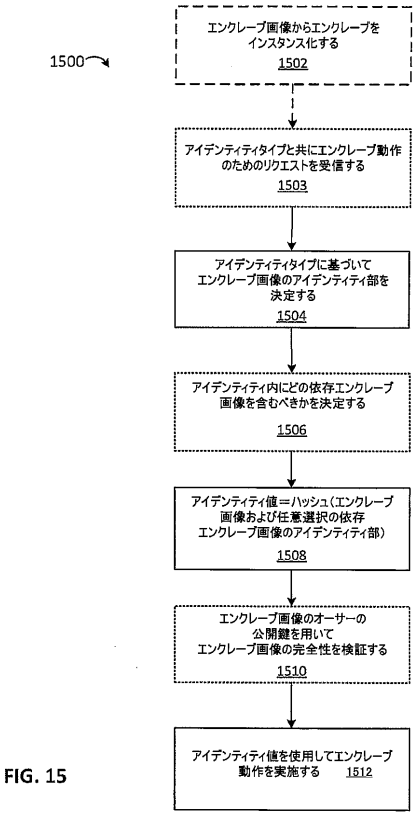


FIG. 15

【図１６】

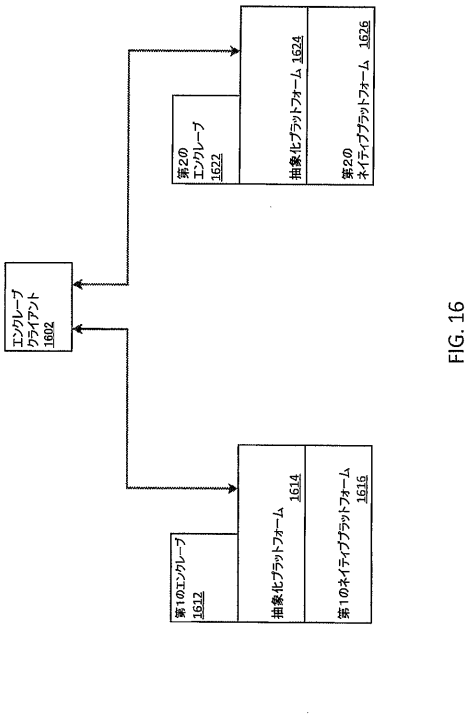


FIG. 16

【図１７】

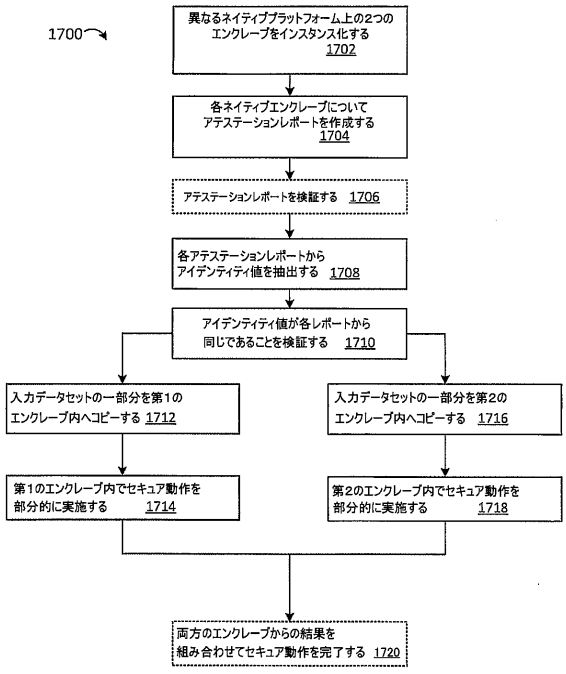


FIG. 17

【図１８】

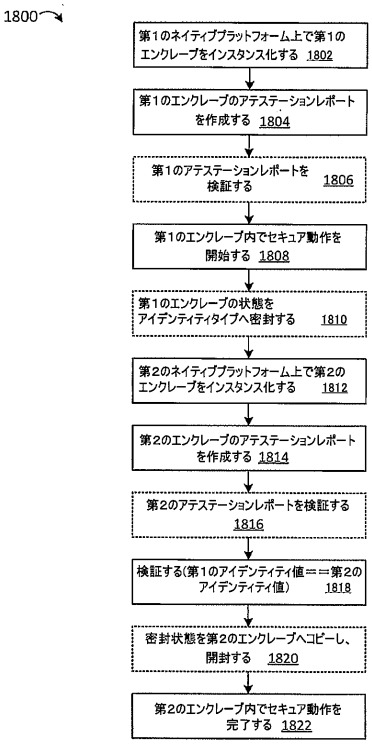


FIG. 18

【図１９】

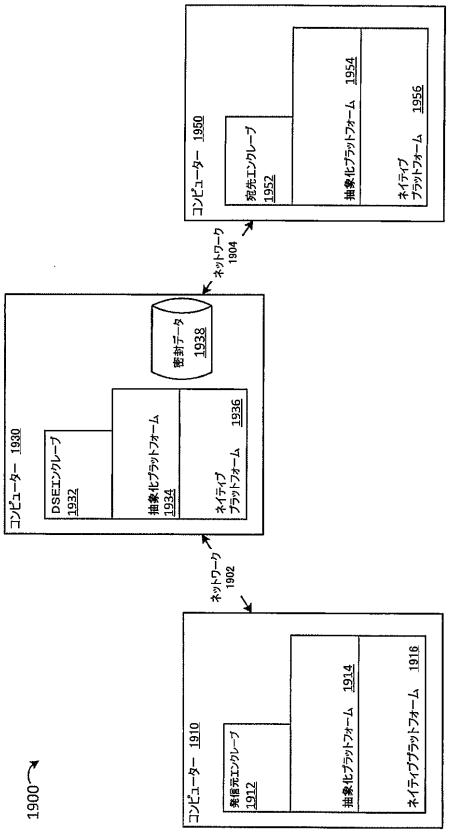


FIG. 19

【図２０】

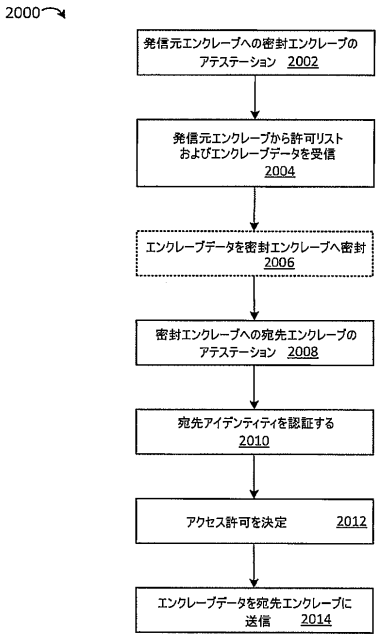


FIG. 20

【図２１】

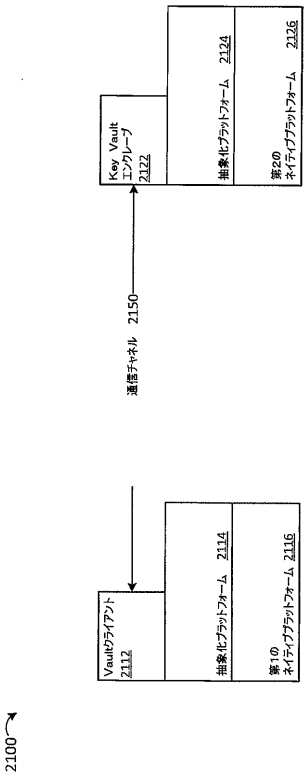


FIG. 21

【図２２】

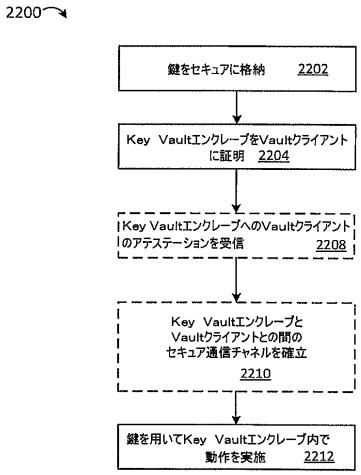


FIG. 22

【図２３】

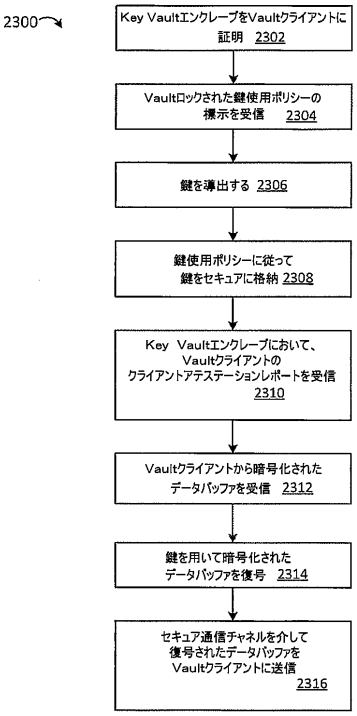
2300
Key VaultエンクレーブをVaultクライアントに証明 2302
Vaultロックされた鍵使用ポリシーの標示を受信 2304
鍵を導出する 2306
鍵使用ポリシーに従って鍵をセキュアに格納 2308
Key VaultエンクレーブにおいてVaultクライアントのクライアントアテステーションレポートを受信 2310
Vaultクライアントから暗号化されたデータバッファを受信 2312
鍵を用いて暗号化されたデータバッファを復号 2314
セキュア通信チャネルを介して復号されたデータバッファをVaultクライアントに送信 2316

FIG. 23

【国際調査報告】

## INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/067451

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/57 G06F21/74
ADD.

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)
G06F

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | John M. , Benjamin O.: "Intel Software Guard Extensions Tutorial Series: Part 1, Intel SGX Foundation", Intel , 7 July 2016 (2016-07-07), XP002779031, Retrieved from the Internet: URL:https://software.intel.com/en-us/articles/intel-software-guard-extensions-tutorial-part-1-foundation [retrieved on 2018-03-12] | 1-3,5, 7-13,15 |
| A | page 5 - page 6 | 4,6,14 |

-----

-/--

[X] 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 | Date of mailing of the international search report |
|---|---|
| 13 March 2018 | 26/03/2018 |

| Name and mailing address of the ISA/ | Authorized officer |
|---|---|
| European Patent Office, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040,<br>Fax: (+31-70) 340-3016 | Mezödi, Stephan |

1

Form PCT/ISA/210 (second sheet) (April 2005)

## INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/067451

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | Anonymous: "Intel Software Guard Extensions Programming Reference", Ref. 329298-002US, 1 October 2014 (2014-10-01), XP055390787, Retrieved from the Internet: URL:https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf [retrieved on 2017-07-13] | 1-3,5, 7-13,15 |
| A | pages 2-14, paragraph 2.17 pages 5-84 | 4,6,14 |
| | ----- | |
| Y | Rebekah Leslie-Hurd: "Sealing and Attestation in Intel Software Guard Extensions (SGX)", Real World Crypto 2016, 8 January 2016 (2016-01-08), XP055458283, Stanford, CA, USA Retrieved from the Internet: URL:https://rwc.iacr.org/2016/Slides/Sealing%20and%20Attestation%20in%20SGX.pdf [retrieved on 2018-03-12] | 1-3,5, 7-13,15 |
| A | page 5 - page 6 | 4,6,14 |
| | ----- | |

1

フロントページの続き

(81)指定国・地域 AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IR,IS,JO,JP,KE,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,LS,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,SM,ST,SV,SY,TH,TJ,TM,TN,TR,TT

(74)代理人 100119781
弁理士 中村 彰吾
(72)発明者 コスタ, マヌエル
アメリカ合衆国 ワシントン州 ９８０５２－６３９９ レッドモンド ワン マイクロソフト ウェイ, マイクロソフト テクノロジー ライセンシング, エルエルシー